UNCLASSIFIED

AD 297 066

Reproduced
by the

ARMED SERVICES TECHNICAL INFORMATION AGENCY
ARLINGTON HALL STATION
ARLINGTON 12, VIRGINIA

UNCLASSIFIED

NOTICE: When government or other drawings, specifications or other data are used for any purpose other than in connection with a definitely related government procurement operation, the U. S. Government thereby incurs no responsibility, nor any obligation whatsoever; and the fact that the Government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use or sell any patented invention that may in any way be related thereto.

N-63-2-5

CATALOGED BY ASTIA
AS AD No. 297066

297 066

ANALYSIS AND SYNTHESIS OF SEQUENTIAL SWITCHING CIRCUITS

by

Zvi Kohavi

Research Report No. PIBMRI-1090-62
for

The Air Force Office of Scientific Research
The U. S. Army Research Office
The Office of Naval Research
Grant No. AFOSR-62-295
June 1963

FEB 27 1963
TISIA A

MRI

POLYTECHNIC INSTITUTE OF BROOKLYN
MICROWAVE RESEARCH INSTITUTE
ELECTRICAL ENGINEERING DEPARTMENT

Research Report No. PIBMRI-1090-62
Grant No. AFOSR-62-295

ANALYSIS AND SYNTHESIS OF SEQUENTIAL SWITCHING CIRCUITS

by

Zvi Kohavi

Polytechnic Institute of Brooklyn
Microwave Research Institute
55 Johnson Street
Brooklyn 1, New York

Grant No. AFOSR-62-295
Research Report No. PIBMRI-1090-62

June 1963

Zvi Kohavi

Zvi Kohavi
Jr. Research Fellow

Approved by:

[illegible]

Edward Smith
Professor

Prepared for

The Air Force Office of Scientific Research
The U.S. Army Research Office
The Office of Naval Research

ACKNOWLEDGMENT

This project was sponsored by the Air Force Office of Scientific Research of the Office of Aerospace Research; the Department of the Army, Army Research Office; and the Department of the Navy, Office of Naval Research under Grant No. AFOSR-62-295.

ABSTRACT

This report deals with two of the most difficult problems in the synthesis of the sequential switching circuits: the problem of simplifying the flow tables of completely and incompletely specified switching functions; and the problem of assigning the secondary variables to the different states of the circuit.

A graphical method is developed for the reduction of the number of states in the sequential machine. The method simplifies the techniques of finding the closed sets of compatibles which from the new set of states of the reduced machine.

The last chapter includes further development of existing methods for the secondary assignment for both synchronous and asynchronous sequential machines. A discussion is also given on the decomposition of sequential machines and its relations to the secondary assignment.

TABLE OF CONTENTS

I. INTRODUCTION

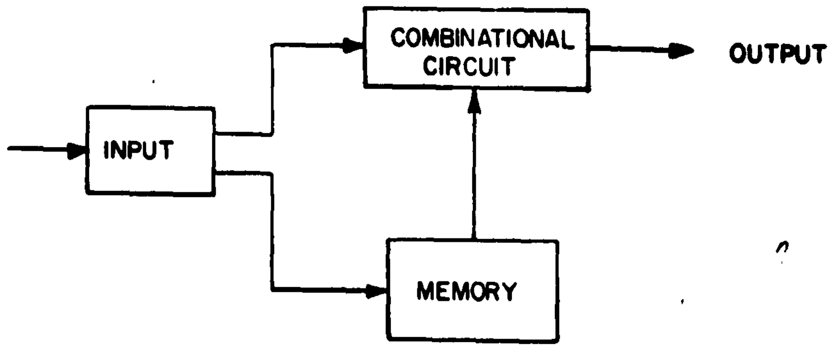
The analysis and synthesis of sequential machines is a branch of switching theory.

A sequential machine is one which contains some memory devices. In the type of combinational circuits, each combination of input variables P_i results in a corresponding output Z_i, the order in which the combination occured or the past combinations do not influence the result.

In the sequential machines the outputs depend on the input-combination and on the present state of the machine. The "present state" (P.S.) of sequential machine is determined by the states of the internal memory elements, which depend on the past history of the machine and on the order of the input combinations.

For a given combination of input variables the output is not unique, it depends on the sequence in which the input variables occur.

In general the block diagram of a sequential circuit is:

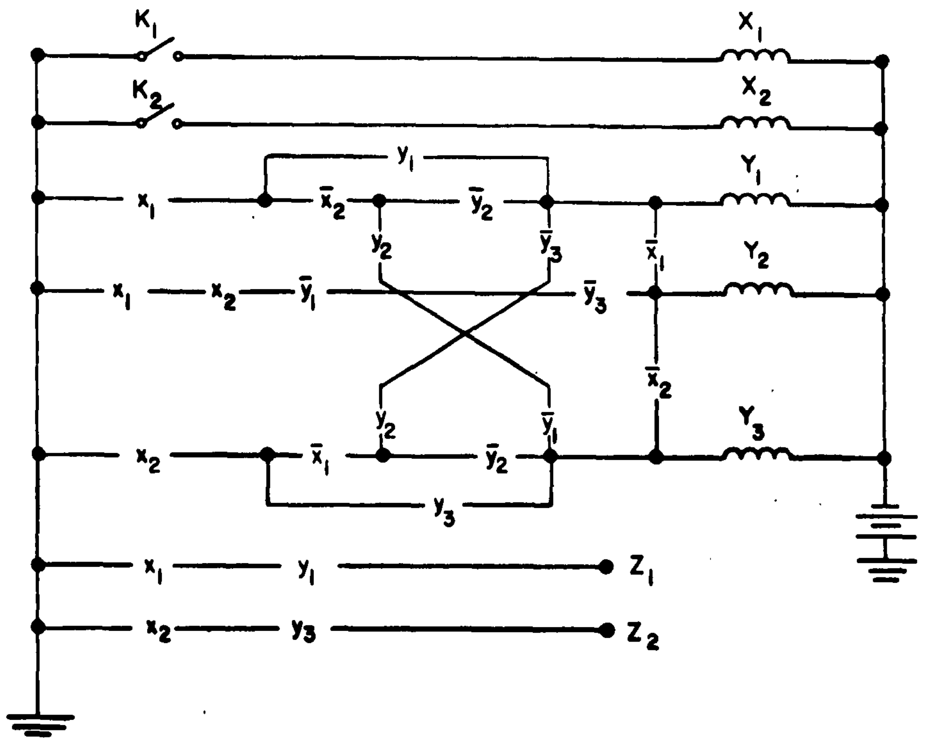


In the synchronous circuits there is an additional clock-pulse which controls all transitions between states. The major part of the thesis is concerned with the synthesis of sequential machines, because the analysis is not entirely obvious but on the whole self-explanatory. The most difficult part of the analysis is simplifying the equations of the circuit, however this is already a part of the systhesis of the sequential machines.

The machines consist of primary relays (or electronic devices) which are all under the immediate control of the input variables. On the other hand, the control network of the secondary elements depends on the change of the input variables and the state of the primary elements. (In Chapter V some methods to reduce this dependency were described.)

Given a network, the analysis procedure is to find the tie sets which give transmission 1 to the memory elements and to the outputs.

Consider the following network



X_1 and X_2 are the primary relays. Y_1, Y_2 and Y_3 are the secondary relays. Z_1 and Z_2 are the outputs.

The tie sets for this network is:

$$Y_1 = x_1(y_1 + \bar{x}_2\bar{y}_2) + x_2\bar{x}_1 y_2\bar{y}_3$$

$$Y_2 = x_1 x_2 \bar{y}_1 \bar{y}_3$$

$$Y_3 = x_2(\bar{x}_1\bar{y}_2 + y_3) + x_1\bar{x}_2 y_2\bar{y}_1$$

$$Z_1 = x_1 y_1$$

$$Z_2 = x_2 y_3$$

By writing these equations we finished with the analysis of the network, however in order to have a better understanding of the operation of the machine we have to simplify these equations (if possible). The simplification procedure is the main subject of this thesis.

The sequential machine is usually described by a matrix which is called "flow-table".

In this thesis we used the flow tables of Huffman[2], Mealy[3], Moore[4], etc. The main idea in all methods is the same, however there are several differences which will be described as follows.

Mealy's model

Input Combination	$I_1\ I_2 \cdots\cdots I_n$	$I_1\ I_2 \cdots\cdots I_n$
Present State	Next State	Output Combination
S_1	$S_{11}\ S_{12} \cdots\cdots S_{1n}$	$Z_{11} \cdots\cdots\cdots Z_{1n}$
S_2	$S_{21}\ S_{22} \cdots\cdots S_{2n}$	$Z_{21} \cdots\cdots\cdots Z_{2n}$
$\vdots$	$\vdots$	$\vdots$
S_m	$S_{m1} \cdots\cdots\cdots S_{mn}$	$Z_{m1}\ Z_{m2} \cdots Z_{mn}$

Let $I = \{I_1\ I_2 \cdots\cdots I_n\}$ be a set of inputs,

$$Z = \{Z_{11}\ Z_{12} \cdots\cdots Z_{1n} \cdots\cdots Z_{mn}\}$$

is a set of outputs.

The left-hand column of the flow table (or as Mealy called it truth table) indicates the present state of the machine and the entries of the table give the corresponding next state to which the machine goes after I_k was applied. For example, if the machine was in present state S_2 and I_n is applied the machine goes to next state S_{2n} and the corresponding output is Z_{2n}.

Moore's model

Moore and Huffman associate each output with a present state of the machine and not with the transition from one state to the other.

	Inputs	
P.S.	$I_1\ I_2 \cdots\cdots I_n$	Output
S_1	$S_{11}\ S_{12} \cdots\cdots S_{1n}$	Z_1
S_2	$S_{21} \qquad S_{2n}$	Z_2
.	.	.
.	.	.
.	.	.
S_m	$S_{m1} \cdots\cdots S_{mn}$	Z_m

Let $S = \{S_1\ S_2\ \ldots\ S_m\}$ be a set of internal states of the machine M_1 let $I = \{I_1 \ldots I_n\}$ be a set of inputs , and $Z = \{Z_1\ \ldots\ Z_m\}$ be the set of outputs which corresponds to S.

The flow-table of Huffman is described in Chapter II. Paull & Unger[10] made minor changes in Mealy's model, however I changed the notation to Mealy's notation in the examples that were taken from their paper.

Chapter II of the thesis is mainly a review of some known design methods. However it is emphasizes what are the achievements of each of the methods described and what are the limitations.

Chapter III developes new methods for the simplification of incompletely specified flow tables and for finding the closed set of compatibles.

Chapter IV deals with the synchronous circuits and Chapter V with the problem of the secondary assignment.

The thesis includes many examples, some of them taken from the references (with some minor changes), the rest are original.

An attempt has been done to solve some of the problems involved in the synthesising of sequential machines, however, even though the main problems are still unsolved they were pointed out and presented.

II. REVIEW OF SOME DESIGN METHODS

A. Huffman's Method.

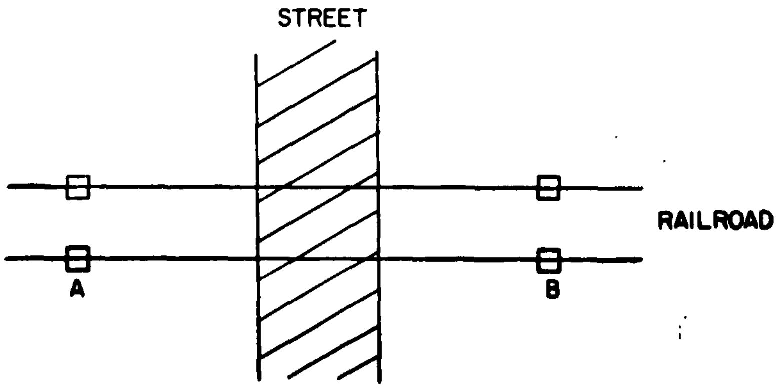
This method presented by Huffman in 1953-54 is good for asynchronous circuit. The technique of synthesising sequential machines consists of the following steps:

1. Description of the machine.
2. Rewriting the description in a form called Primitive Flow Table. (P.F.T.)
3. Eliminating equivalent states in the P.F.T.
4. Drawing of a merger diagram in order to get the Reduced Flow Table. (R.F.T.)
5. Assigning the secondary states in order to represent the rows of the R.F.T.
6. Writing the exitation and output matrices which turn the sequential problem into a combinational one.

The method will be illustrated by several examples which will emphasis the process through all steps.

Ex. 2.1

At a junction of a one-line railroad and a street, traffic lights are needed. The lights will be controlled by the trains. When the train is 1500 feet from the intersection, the lights have to change from green to red until the train is 1500 ft. after the intersection.



At the points A and B (each of which is 1500 ft. from the junction) we shall put two contacts which will form the input contacts of the circuit.

If we give decimal weight to the contacts A = 1 B = 2, we can describe the same system in a numerical method.

X	0 1 0 2 0	0 1 0 1 0	0 2 0 1 0	0 2 0 2 0
Z	0 1 1 1 0	0 1 1 1 0	0 1 1 1 0	0 1 1 1 0

X - is the input contact.

Z - is the output (Z = 1 means red light on, Z = 0 - red light off).

The explanation to this description is simple; the initial state is X = 0 (no train near the junction) Z = 0. Next state, a train approaches the junction from the left; switch A is pressed (A = 1), and red light is on - Z = 1. The third state is after the train leaves point A but didn't yet arrive to point B - no switch is pressed, but red light is still on. In state 4, switch B is pressed, the lights are still on till the fifth state which is equal to the first one. The third sequence is equivalent to the first one. The second and fourth sequences take in account a case when the train pressed switch A or B, but before pressing switch B (or A), it backs to the same direction from which it has just come.

This problem is not a practical solution for traffic lights, it is just an example of synthesising sequential switching circuits.

The numerical description of the problem is not used by Huffman, and it is not always necessary to use it, but in most cases it simplifies the procedure of writing the flow table.

Step 2.

B A				
0	1	3	2	Z
0 0	0 1	1 1	1 0	
①	2	-	6	0
3	②	-	-	1
③	5	-	4	1
1	-	-	④	1
1	⑤	-	-	1
3	-	-	⑥	1

The numbers 0, 1, 2, 3 are the decimal weight of 00, 01, 10.11. Column 3 is not specified because switches A and B can never be pressed together. (Assuming the length of the train never exceeds 3000 ft.).

The P.F. T. follows directly from the numerical description of the problem.

① is the initial state x = 0 Z = 0 from x = 0 we move to x = 1 Z = 1 (state ②). The 2 indicates an unstable state, which occurs immediately after pressing switch A, but before the internal element changed. The 2 becomes stable when it moves vertically to ② .

In the same manner the whole table is filled.

Step 3.

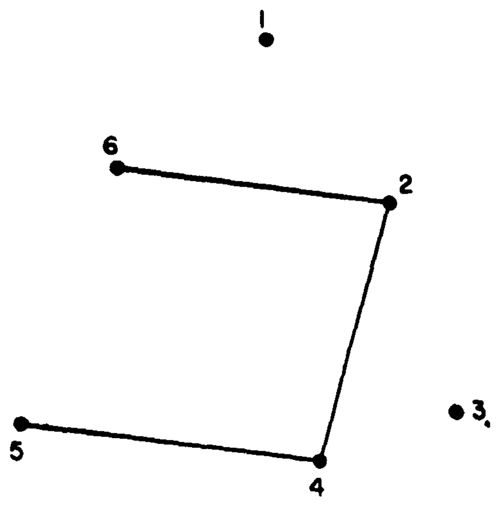
The conditions for equivalency of rows are as follows.

Two rows are equivalent if their outputs are the same (or "are not in contradiction" in case that Z is not completely specified), and there is no contradiction between their internal entries. The rows are numbered correspondingly to the stable states.

Therefore in our example there are no equivalent states because ② and⑤ and ④ - ⑥ have 1, 3 as unstable states, but 1, 3 have different outputs; therefore, they are not equivalent.

Step 4.

An array of points is drawn to correspond with the states in the P.F.T. (After rule 3 was applied on it.) Rows may be merged if there is no contradiction between their entries.



The states that form the R.F.T. are (1) (2, 6) (3) (4, 5)

	BA 0 0	0 1	1 1	1 0
a	①	2	-	6
b	3	②	-	⑥
c	③	5	-	4
d	1	⑤	-	④

Steps 5, 6 will be discussed in general in Chapter V. The results however are:

$0\ 0 \rightarrow a$

$0\ 1 \rightarrow b$

$1\ 1 \rightarrow c$

$1\ 0 \rightarrow d$

The exitation matrix is

$Y_2Y_1 \backslash X_2X_1$	0 0	0 1	1 1	1 0
0 0	0 0	0 1	-	0 1
0 1	1 1	0 1	-	0 1
1 1	1 1	1 0	-	1 0
1 0	0 0	1 0	-	1 0

$$Y_1 = \overline{y}_2 \; (x_1 + x_2) + y_1 \overline{x}_2 \overline{x}_1$$

$$Y_2 = y_2 \; (x_1 + x_2) + y_1 \overline{x}_2 \overline{x}_1$$

The output matrix is written from the P.F.T.
Z = 1 for states 2, 3, 4, 5, 6.

$Y_2Y_1 \backslash x_2x_1$	0 0	0 1	1 1	1 0
0 0	0	-	-	-
0 1	-	1	-	1
1 1	1	-	-	-
1 0	-	1	-	1

$$Z = y_1 + y_2$$

Ex. 2.2

A two inputs, two outputs, sequential circuit has the following properties:

1. The inputs variables are to change one at a time.
2. After any change of the input state the output is always either the initial input state or the final output state.
3. When the input states are represented in binary form, an input change which increases the decimal value of the binary input causes an output of either 0 0 or 1 1. If the decimal value decreases, the output is either 0 1 or 1 0.

This problem will be best illustrated by the following numerical description.

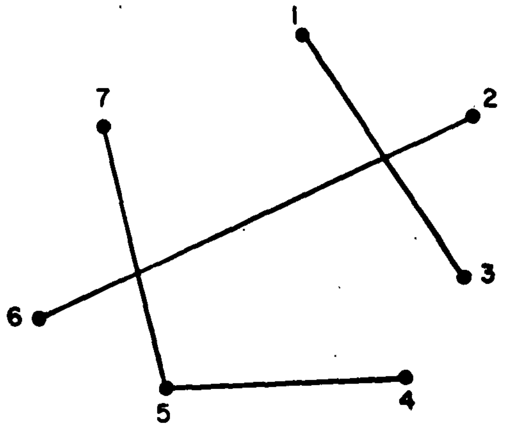
	P. S.	N. S.	Output
1	0 0	0 1	0 0
2	0 0	1 0	0 0
3	0 1	0 0	0 1
4	0 1	1 1	1 1
5	1 1	0 1	0 1
6	1 1	1 0	1 0
7	1 0	0 0	1 0
8	1 0	1 1	1 1

P. S. - Present State
N. S. - Next State

Each input state has only two ways to change, therefore under column P.S. each state is written twice. In column N. S. the corresponding changes are written and in the third column the corresponding output is given. For example 0 1(decimal value one) can change to 0 0 or 1 1. A change to 0 0 decreases the decimal value from one to zero hence the output may be 0 1 or 1 0, but as the input was 0 1, the output according to property 2 is 0 1. A change to 1 1 increases the decimal value to three, hence the output may be either 0 0 or 1 1, as the final output state is 1 1, the output is 1 1. The flow table can be derived directly by noticing that each input state has two different stable states. A check for equivalent rows will result in combining rows 4 and 8 as both are in the same state 11 and their output is the same 1 1.

x_2 x_1				z_2z_1
0 0	0 1	1 1	1 0	
①	3	-	6	0 1
②	3	-	6	1 0
1	③	5	-	0 0
1	④	5	-	0 1
-	4	⑤	7	1 1
2	-	5	⑥	0 0
2	-	5	⑦	1 0

The merging diagram is



The solution may be:

(1, 3) (2, 6) (4, 5) (7)

or

(1, 3) (2, 6) (4) (5, 7)

The corresponding R. F. T. are:

1)

①	③	5	6
②	3	5	⑥
1	④	⑤	7
2	-	5	⑦

2)

①	③	5	6
②	3	5	⑥
1	④	5	-
2	4	⑤	7

There is no way to determine which flow table is better to realize, therefore, an arbitrary choice is done to realize the first flow table.

From the R.F.T. we have to find the exitation matrix, hence to assign the secondary states. In order to assign the secondary states to the states of the flow table let us rewrite the R.F.T. in such a way that each unstable state will be adjacent to the corresponding stable state, assuming that the R.F.T. is a map of Karnaugh.

①	③	5	6
②	3	5	⑥
2	-	5	⑦
1	④	⑤	7

A check of the R.F.T. shows that the unstable state 5 in the second row-third column is not near its stable state ⑤; in this case we have a race condition. The problem of races is explained widely in Chapter V, and as this race is not critical, we shall leave it without any further explanation.

The transition and output matrices are:

$y_2y_1 \backslash x_2x_1$	0 0	0 1	1 1	1 0
0 0	0 0	0 0	1 0	0 1
0 1	0 1	0 0	1 0	0 1
1 1	0 1	-	1 0	1 1
1 0	0 0	1 0	1 0	1 1

<u>output</u>

$y_2y_1 \backslash x_2x_1$	0 0	0 1	1 1	1 0
0 0	0 1	0 0	1 0	0 1
0 1	1 0	1 0	1 0	0 0
1 1	1 0	1 0	1 0	1 0
1 0	0 1	0 1	1 1	0 1

While assigning the unstable entries to the output matrix, we assume that the response time of the network to the output signals is long enough, compared to the duration of the unstable states. Therefore, we assign the entries in such a way that the output network will be as simple as possible.

$$Y_1 = x_2 \bar{x}_1 + y_1\bar{x}_1 = \bar{x}_1 (x_2 + y_1)$$

$$Y_2 = x_1x_2 + x_2y_2 + x_1y_2 = x_2 (x_1 + y_2) + x_1y_2$$

$$Z_1 = \bar{x}_1\bar{y}_1 + \bar{y}_1\bar{y}_2 = \bar{y}_1 (\bar{x}_1 + \bar{y}_2)$$

$$Z_2 = \bar{x}_2 y_1 + x_1x_2 + y_1y_2 = y_1 (\bar{x}_2 + y_2) + x_1x_2$$

Ex. 2.3

Find a R.F.T. for the following flow table.

x_1x_2					
0 0	0 1	1 1	1 0	Z_1	Z_2
8	2	7	①	1	0
4	②	7	6	1	-
8	2	③	1	0	-
④	9	7	6	0	0
4	⑤	3	6	1	1
8	5	3	⑥	-	0
8	5	⑦	1	-	1
⑧	2	7	6	0	0
8	⑨	7	6	1	0

A brief search will indicate that 4 and 8 will be equivalent if 2 and 9 are; 2 and 9 will be if 4 and 8 are; therefore, as they depend on each other, they can form two equivalent sets (4, 8) (2, 9).

At this point it is worthwhile to note that according to Huffman's method once a set of rows is combined, its output is fixed; hence the output of row No. 2 becomes 10 as the output of row No. 9. From this we conclude **that row 5 cannot be combined** with any other row. Rows 3, 7 depend on the **equivalence of 2 & 5; hence they cannot be combined,** same for rows 1, 6.

The new flow table will have 7 rows:

(1) (2, 9) (3) (4, 8) (5) (6) (7)

However one can see that the set of rows

(1, 6) (2, 5) (3, 7) (4) (8) (9)

is also a solution for this problem. From this **example we notice that the first** assignment 2 = 9 **enabled us to find the minimal solution.**

Ex. 2.4

Given the following flow table:

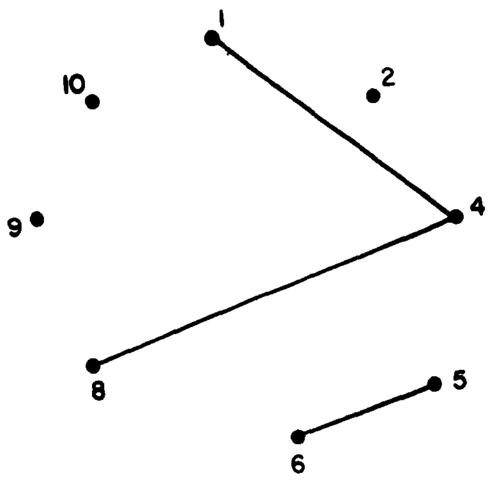
0 0	0 1	1 1	1 0	Z
①	2	7	8	0 -
3	②	-	6	1 1
③	2	4	8	0 1
-	-	④	-	1 -
⑤	2	7	6	- 0
5	-	-	⑥	0 1
1	-	⑦	8	- 0
1	9	-	⑧	- 1
-	⑨	-	10	1 1
1	2	-	⑩	0 -

In order to reduce this flow table according to Huffman's method, we have to find redundant stable states by checking all states which belong to the same input conditions (i.e., that are in the same column). ①and③may be equivalent if④and⑦ are. ④and⑦are equivalent. ①and⑤cannot be equivalent because they contradict in their output. (The output of①after merging it with state③is now 01.)

②and⑨are equivalent if⑥and ⑩ are. ⑥and ⑩ if①and⑤are, hence they cannot be combined. The following flow table cannot be reduced anymore.

0 0	0 1	1 1	1 0	Z
①	2	4	8	0 1
1	②	-	6	1 1
1	-	④	8	1 0
⑤	2	4	6	- 0
5	-	-	⑥	0 1
1	9	-	⑧	- 1
-	⑨	-	10	1 1
1	2	-	⑩	0 -

The merger diagram is:



The merged flow table consists of 6 states (1) (2) (4, 8) (5, 6) (9) (10) which require 3 memory elements.

However the combining of states ① and ③ was not a good decision as is shown in the following flow table.

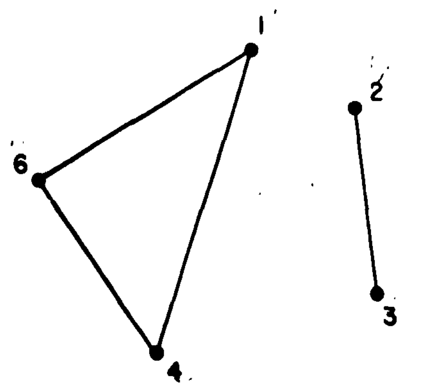
Assuming ① and ③ are not combined we get:

① and ⑤ are equivalent if ⑥ and ⑧ are. ⑥ and ⑧ if ① and ⑤; hence both pairs may be combined. In the same manner we get that ② and ⑨, ④ and ⑦, ⑥ ⑧ and ⑩ are equivalent and the reduced flow table is:

0 0	0 1	1 1	10	Z	
①	2	4	6	0	0
3	②	-	6	1	1
③	2	4	6	0	1
1	-	④	6	1	0
1	2	-	⑥	0	1

This flow table consists of 5 states instead of the 8 states we received after combining ① and ⑤.

The merger diagram is:



(1, 4, 6) = A (2, 3) = B

	0 0	0 1	1 1	1 0
A	A	B	A	A
B	B	B	A	A

The exitation matrix is

$$\begin{array}{c|cccc} Y \backslash x_2x_1 & 00 & 01 & 11 & 10 \\ \hline 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \end{array}$$

$$Y = \bar{x}_2 x_1 + \bar{x}_2 y = \bar{x}_2 (x_1 + y)$$

The output matrix assuming unstable states as "don't cares" is:

$$\begin{array}{c|cccc} Y \backslash x_2x_1 & 00 & 01 & 11 & 10 \\ \hline 0 & 00 & - & 10 & 01 \\ 1 & 01 & 11 & - & - \end{array} \qquad Z_2 \, Z_1$$

$Z_1 = y + x_2 \bar{x}_1$

$Z_2 = x_1$

This solution required only 1 memory element instead of the 3 in the previous reduction. We conclude that one has to be very careful when he decides which rows are to be combined. An unsuccessful choice will result in a flow table which cannot be reduced any more but is far from the optimal solution.

B. The Methods of Moore and Mealy

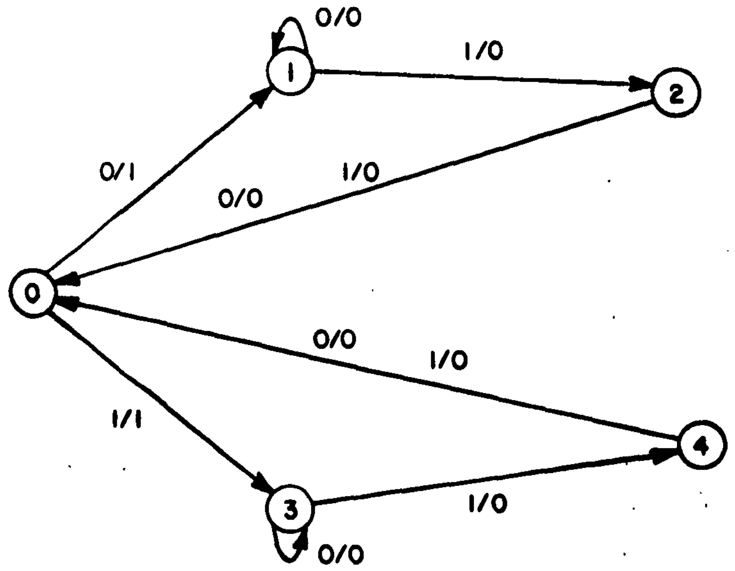
Moore[4] and Mealy[3] in 1955 developed new similar methods for synthesising sequential machines. Both methods were developed specially for synchronous machines, however they hold for asynchronous machines too.

The basic idea of the method is given in this chapter; additional examples and explanations are given in chapter IV.

Both methods use the state diagram (Moore called it "transition diagram") in order to illustrate the operation of the machine. The state diagram is a graph that represents the transition from one state to the other and the flow of information through the machine.

Moore follows the representation of Huffman where the outputs are determined by the states of the machine, while Mealy associates the outputs with the inputs to the machine in each of its states.

As an example consider the following state diagram.

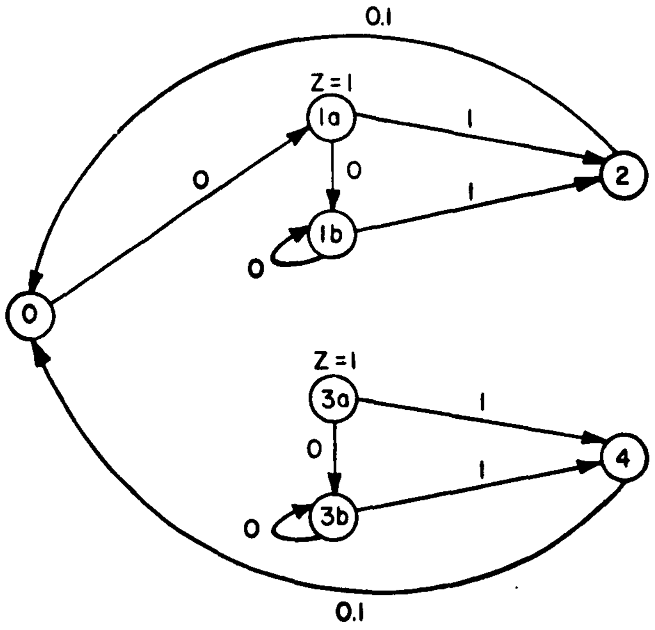


Each node represents a state of the machine; each line represents the transition between the states. Each line is marked to show what input change causes the transition and what is the output response. The arrow indicates the direction of the transition.

The symbol $^{a}/b$ means that the input to the state is "a" and the output is "b". For example $^{0}/1$ means that a zero input causes an output of 1.

This machine can be represented by the methods of Moore or Huffman, but it requires seven states instead of five. States 1 and 3 must be subdivided into 1a 1b and 3a 3b each of which corresponds to a different output position.

1a , 3a corresponds to output of 1 . 0 , 2 , 4 , 1b , 3b corresponds to an output of 0 .



In general, each state in the model of Mealy, the inputs of which "carry" different ouputs, must be subdivided into subsets, each of which corresponds to one output.

The number of substates of each state is equal to the number of different outputs. For example, the inputs to state 0 are 0 and 1 from states 2 and 4 . The outputs that correspond to these inputs are always 0; hence state 0 has one substate that is identical to the state itself.

But state 1 has as inputs $^0/0$ $^0/1$; hence we need 2 substates - one for the position Z = 1 and the other for Z = 0 . The transition lines must be drawn again to correspond to the new states.

As a conclusion, we can see that the method of Moore is similar to that of Huffman - when dealing with asynchronous circuits - therefore we shall emphasise the method of Mealy.

In order to determine state equivalency we separate the states into sets which have the same input - output relations.

States which belong to the same set may be equivalent; states which do not belong to the same set may not be equivalent. Afterwards a check is made to determine whether the transition lines from each set can be merged.

As an example we check the previous state diagram, and we notice that states 1, 2, 3, 4 have the same input - output relations, (i.e. $^0/0$ $^1/0$). The two sets are therefore:

(0) . (1, 2, 3, 4)

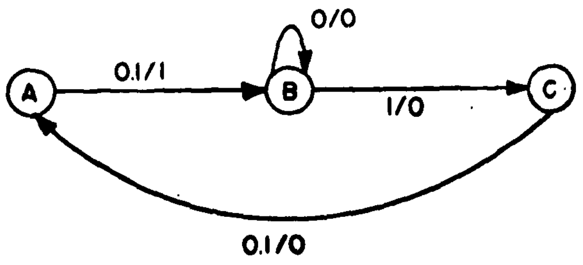
A check for x = 0 is applied to the machine. The results are that from states 2 and 4 the machine goes to state 0 , while from states 1 and 3 the machine does not go anywhere.

Hence we have to subdivide (1 , 2, 3, 4) into (1, 3) and (2, 4). If 1 is applied to states 1 or 3, the machine goes to the set (2, 4) and from which, by applying either 0 or 1, it goes to state 0 .

The final solution is:

A = (0) B = (1, 3) C = (2, 4)

The reduced state diagram is



The state diagram is a graphical way to show the operation of the machine. Same results we could have received from the corresponding flow table.

P.S.	N.S		Output	
	0	1	0	1
0	1	3	1	1
1	1	2	0	0
2	0	0	0	0
3	3	4	0	0
4	0	0	0	0

This flow table suggested by Mealy can be reduced to the following flow table which corresponds to the reduced state diagram as found before.

P.S.	N.S.		Output	
	0	1	0	1
A=0	B	B	1	1
B=1, 3	B	C	0	0
C=2, 4	A	A	0	0

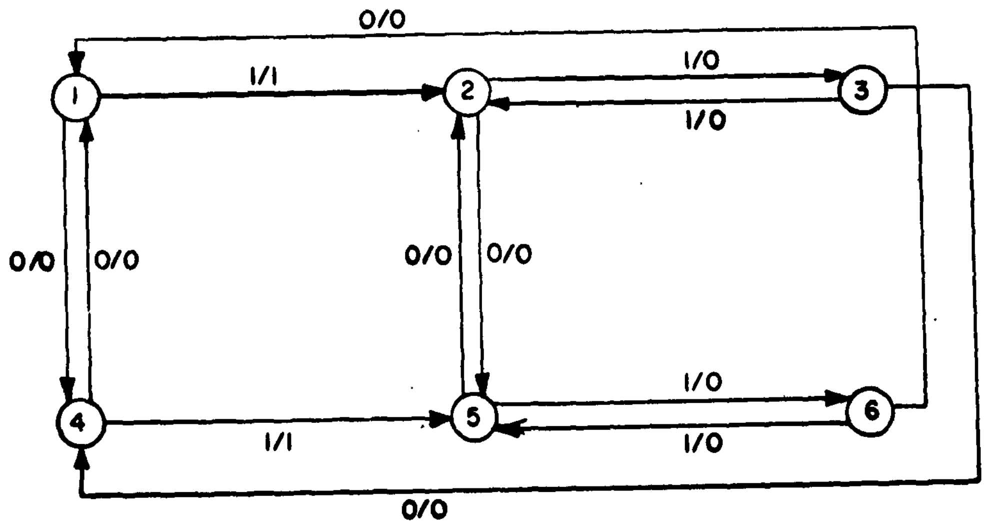
The rules for simplifying flow tables are as follows:

Two states of a machine are said to be equivalent if and only if there is no experiment which can be performed on the machine that will give different output results if the machine was initially in either state.

In other words two states of a machine are equivalent if their outputs do not contradict and a change of any input in both states will cause the machine to move to a common next state. In our example, there is no experiment that can be performed on the machine which will distinguish between state 1 and 3 or 2 and 4 , and their outputs are the same.

Ex. 2.5

Given the following state diagram:



The corresponding flow table is:

P.S.	N.S.		Output	
	0	1	0	1
1	4	2	0	1
2	5	3	0	0
3	4	2	0	0
4	1	5	0	1
5	2	6	0	0
6	1	5	0	0

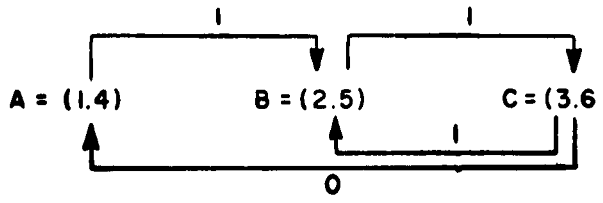
A grouping of the states according to their outputs will result in

(1, 4) (2, 3, 5, 6)

Set (2, 3, 5, 6) implies - when applying to it 0 input - the set (5, 4, 2, 1) but (5, 2) cannot be merged with (4, 1) because of different outputs. Therefore, we have to sub-divide this set and we obtain:

(1, 4) (2, 5) (3, 6)

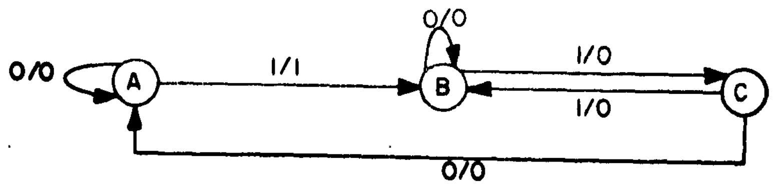
A check shows that these sub sets can form a closed set.



The reduced flow table is:

P.S.	W.S.		Output	
	0	1	0	1
A	A	B	0	1
B	B	C	0	0
C	A	B	0	0

The correspond state diagram is:



In the end of Chapter II. A several examples were given to show that the reduction method of Huffman is not sufficient when the flow tables are not completely specified. Same disadvantage applies to the methods of Moore and Mealy as will be shown in the following examples.

<u>Ex. 2.6</u>

Given the following flow table:

P.S.	N.S.				Output			
IMP →	00	01	11	10	00	01	11	10
1	1	5	-	1	00	01	-	01
2	-	3	2	4	-	10	00	11
3	1	3	-	-	00	10	-	-
4	1	-	-	4	00	-	-	11
5	-	5	6	-	-	01	00	-
6	-	7	6	7	-	10	00	11
7	1	-	-	7	00	-	-	11

According to the rules of Mealey, states, the outputs of which are not contradictory, are candidates for equivalency.

This rule can be applied to the sets:

(1, 5) (4, 7) (2, 3, 6) .

(2, 3, 6) implies (3, 7) , but (3, 7) is not in any set; therefore, we have to divide (2, 3, 6) into (2, 3) (6) .

The closed set is therefore:

A = (1, 5)   B = (4, 7)   C = (2, 3)   D = (6)

P.S.	N.S.				Output			
	00	01	11	10	00	01	11	10
A	A	A	D	A	00	01	00	01
B	A	-	-	B	00	01	00	11
C	A	C	C	B	00	10	00	11
D	-	B	D	B	-	10	00	11

This flow table cannot be reduced anymore. It requires at least two secondary variables.

In the next chapter, ex. 3.7 and 2.9, we shall show that the same flow table can be reduced to two states and only one **secondary variable.**

<u>Ex. 2.7</u> Given the following flow table:

	x_1	x_2	x_3	x_1	x_2	x_3
1	3	5	-	0	1	-
2	3	5	-	0	-	-
3	2	3	1	-	0	-
4	2	3	5	0	-	-
5	-	5	1	-	0	-

According to Huffman's or Mealy's methods, rows 1 and 2 **are equivalent; therefore,** we can immidiately combine them.

1-2	3	5	-	0	1	-
3	2	3	2	-	0	-
4	2	3	5	0	-	-
5	-	5	2	-	0	-

Row 2 can be combined with 4 if 2 and 3 are equivalent, but 2-3 **contradict** in their outputs hence row 2 cannot be combined with any **other row.**

Row 3-4 depend on 2-5, hence they are not equivalent.

Finally we find that rows 3 and 5 are equivalent **and the result is:**

1-2	3	3	-	0	1	-
3-5	2	3	2	-	0	-
4	2	3	3	0	-	-

To realize this R.F.T. we need 2 memory elements. But if we would not **have combined rows 1 and 2 in the beginning a solution that requires only 1 memory**

element could have been found.

A = 1.4	A	B	B	B	0	1	-
B = 2.3.5	B	B	B	A	0	-	-

The disadvantage of these methods is in the fact that once two rows have been merged, the unspecified entries are assigned in such a way that is not always optimal. The methods of Mealy, Moore or Huffman holds for completely specified flow table, however their techniques for reducing incompletely specified flow tables is in general unsatisfactory. A method for reducing flow tables of incompletely specified sequential switching circuits was developed by Paull and Unger[10].

Another method which holds for completely and incompletely specified sequential machines is developed in chapter III.

C. The Method of Paull and Unger

As shown in several previous examples the results received by the methods of Mealy and Huffman were insufficient in cases where the flow table was not completely specified. Paull and Unger[10] in their paper developed a technique to minimize the number of states in incompletely specified sequential switching circuits. In order to explain the method some of the main definitions and theorems are given as follows:

A grouping of all the rows of a flow table into c sets will be said to be closed, if every set implied by any c_i is included in one of the c sets. "A compatible is a set of states of a flow table that constitutes one member of some closed grouping" i.e. states that do not contradict in their outputs may form a compatible if the corresponding entries belong to any set of states in the flow table. A set of states is a compatible if, and only if every pair of states in that set is a compatible. (i.e. (1,2,3) are compatible only if (1,2) (2,3) and (1,3) are compatible.

According to this method one must find all possible Mc's from which we may choose the lower bound of compatibles that will form the R.F.T. After choosing the lower bound of the compatibles a test for closure is done. If the set of compatibles is closed the R.F.T. can be constructed. If it is not closed tests for closure are done after adding more compatibles to the set.

The procedure of finding the compatibles is carried out in an "implication table" as illustrated in the following example.

P.S.	N.S.				Output			
Inp →	00	01	11	10	00	01	11	10
1	-	3	5	2	-	1	1	1
2	5	-	-	-	0	-	-	-
3	6	6	-	-	0	1	-	-
4	-	-	2	-	-	-	1	-
5	-	6	1	4	-	0	0	1
6	3	-	2	3	0	-	0	1

Flow table.

2	V				
3	36	56			
4	25	V	V		
5	X	V	X	X	
6	X	35X	V	X	12 34
	1	2	3	4	5

Implication table.

In the implication table each square corresponds to a pair of states in the flow table.

The filling of the implication table is as follows: If for row-pair ab, the outputs are not contradictory then enter all row pairs implied by ab in the ab square. (Square (13) (14) etc. in our example). If no pairs are to be entered a check mark is inserted. (Square (12) (24) etc.) If the outputs of rows ab contradict an X is placed in the corresponding square. (cells (15) (16) etc.).

After the table is filled a check is done if there is no contradiction to the entries of the table. The check in quare (13) which has the entries (36) shows that (36) are checked, therefore we consider cell (13) as a checked cell. A test of the entries of cell (26) shows that the entries (35) are crossed in the square (35), therefore an X is placed in cell (26) and the cell is ignored as if it was a "crossed" cell.

The highest order MC is in column 1 (123456), but (15) and (16) contradict therefore we divide this compatible to two compatibles,

(1234) (23456).

A check in colum 2 shows that (26) contradict, therefore we divide the second compatible and we obtain:

(1234) (2345) (3456)

In columm3 we observe that (35) contradict and we get:

(1234) (234) (245) (346) (456)

As the second compatible is included in the first we ignore it. From the fourth column we get:

(1234) (24) (25) (34) (36) (4) (56) or (1234) (25) (36) (56)

As there are no X's in the fifth colum this is the answer, this is the set of the MC's. Paull and Unger suggested another equivalent technique to find the MC's by starting with the smallest compatible as found in the last column and expanding it according to the previous columns. The results are as follows:

1) (56)
2) (34) (36) (56)
3) (234) (36) (56) (25)
4) (1234) (36) (56) (25)

The upper bound is now 4 states, but the lower bound is only two states:

(1234) (56)

A check if the lower bound is closed shows that (1234) implies (56) (36) and (25), as (36) and (25) are not members of any compatible the lower bound cannot construct the R.F.T. A check shows that (25) and (36) are closed therefore we need only rows 1 and 4 which form a compatible and imply (25) which is already included in the set of compatibles, hence the R.F.T. consists of 3 states:

		00	01	11	10	00	01	11	10
A = (25)	A	-	C	B	B	-	1	1	1
B = (36)	B	B	C	A	A	0	0	0	1
C = (14)	C	C	C	B	C	0	1	0	1

Ex. 2.8

Reduce the following flow table:

Inp.	00	01	11	10	00	01	11	10
P.S.	N. S.				Output			
1	1	2	-	-	00	01	-	-
2	-	2	3	-	-	01	00	-
3	-	-	3	4	-	-	00	00
4	5	-	-	4	00	-	-	00
5	5	-	-	6	00	-	-	00
6	-	-	7	6	-	-	10	00
7	-	8	7	-	-	00	10	-
8	1	8	-	-	00	00	-	-

2	V						
3	V	V					
4	1.5	V	V				
5	V	V	4.6	4.6			
6	V	X	X	V	V		
7	2.8 X	X	X	V	V	V	
8	X	X	V	V	V	V	V
	1	2	3	4	5	6	7

1) (12345678)

2) (123456) (2345678)

3) (12345) (13456) (2345) (345678)

4) (12345) (1456) (45678) (3458)

5) (12345) (1456) (45678) (3458)

The upper bound is 4 compatibles the lower bound is 2 compatibles:

(12345) (45678)

A check for closure shows that (1 5) and (46) are implied and both are included in the compatibles.

As state 5 is included in the first compatible it is redundent in the second one. Same rule cannot be applied to State 4 as the set (46) is applied by the first compatible. The R.F.T. will consist of two states:

A = (12345)

B = (4678)

	N. S.				Output			
	00	01	11	10	00	01	11	10
A	A	A	A	A	00	01	00	00
B	A	B	B	B	00	00	10	00

Ex. 2.9.

Consider now the flow table of example 2.6. According to the method of Paull and Unger we get:

2	X					
3	X	V				
4	X	V	V			
5	V	X	X	V		
6	X	4.7 3.7 V	3.7 V	4.7 V	X	
7	X	4.7 V	V	V	V	V
	1	2	3	4	5	6

1) (1, 5) (2, 3, 4, 5, 6, 7)

2) (1, 5) (2, 3, 4, 6, 7) (3, 4, 5, 6, 7)

3) (1, 5) (2, 3, 4, 6, 7) (3, 4, 6, 7) (4, 5, 6, 7)

4) (1, 5) (2, 3, 4, 6, 7) (4, 5, 7)

The lower bounds is closed, hence it is the solution.

A = (1, 5) B = (2, 3, 4, 6, 7)

D. The Design of the Output Circuit.

According to Huffman's method the outputs correspond to the states of the machine, hence for all stable states the outputs are well defined. By "defined" we mean that the output may be defined as a one, zero, or don't care. The unstable states however, are in general not specified directly, and so, we have the freedom to assign to those entries a zero or a one in order to get most economic circuit. There are several limitations for such an assignment, these limitations may be divided into two groups:

The first group consists of general limitations which hold for almost all networks. The second group consists of restrictions which are specified for each network.

The first general principle is not to allow any false output while the machine changes from one stable state to another stable state, or in other words no output is allowed to change its value during the transient time for other purpose than to get the newly assigned value. This principal can be explained by the following examples.

Machine A has in state $\textcircled{1}$ an output 00 , in state $\textcircled{2}$ the output is the same. While changing $\textcircled{1} \rightarrow 2 \rightarrow \textcircled{2}$ it is not desirable that in the nonstable state 2 the output will be different from 00 . This is correct for all cases except where the output has a very long time constant compared to the time constant of the nonstable states in the machine.

Machine B has in state $\textcircled{1}$ an output 01 and in state $\textcircled{2}$ an output 11 . It is not desirable to assign to the nonstable state 2 an output 10 or 00 . The first output changes from 0 to 1 therefore the nonstable state may be a 0 or an one. Till the change $Z_1 = 0$ and it may remain so during the nonstable state or it may change to $Z_1 = 1$ and to remain so during the nonstable state and during state $\textcircled{2}$. However $Z_2 = 1$ in both states hence it is not allowed to change Z_2 momentarily from 1 to 0 and back to 1 . The possibilities of assigning the nonstable states in this machine are 01 or 11 . In this case too, if the response time of Z_2 is long we may assume that the change of Z_2 in state 2 does not cause any disturbance to the machine.

From the same reasoning we conclude that if Z in state $\textcircled{1}$ is 01 and in state $\textcircled{2}$ 11 , there are two possibilities to assign Z in state 2 , 01 or 11 . The last case is machine C where Z in state $\textcircled{1}$ is 01 and in state $\textcircled{2}$ 10 . The question is if we are allowed to assign $Z = 00$ or $Z = 11$ in the unstable state 2 . Z_1 and Z_2 change from 0 to 1 or from 1 to 1 , hence it is allowed to assign to Z_1 and Z_2 0 or 1 . Therefore in this case all possible assignments of the outputs are allowed. As a note we may add that these reasons are not strong as rules, and the decision is always left to the designer. Caldwell' in his book states that in the example of machine C assigning to Z the values of 00 or 11 will cause a momentary false output. However as explained before these assignments are correct because no output changes - twice during the transient time, and all the changes are necessary in order to get the values of state ② .

Consider machine D of example 2.1 . In this machine the requirement is that all output changes are to occur as soon as possible.

$Y_2\ Y_1$ \ $X_2\ X_1$	00	01	11	10
00	①	2	-	6
01	3	②	-	⑥
11	③	5	-	4
10	1	⑤	-	④

flow table

	00	01	11	10
00	0		-	
00		1	-	1
11	1		-	
10		1	-	1

Z map

To states 4 and 5 we have to assign the value Z = 1 because to these unstable states the machine comes from state ③ in which Z = 1 , and from these states the machine goes to states ④ and ⑤ in which Z = 1 . Same for entry 3 . In entries 2 and 6 we have the choice to assign Z = 0 or Z = 1 , but as we want that any changes in Z will occure as soon as possible we assign Z = 1 . Same reason gives in state 1 Z = 0 .

0	1	-	1
1	1	-	1
1	1	-	1
0	1	-	1

Final Z table

If the requirement was that the green light is to change to red as soon as possible but red into green as late as possible (i.e. a change from Z = 0 to Z = 1 as soon as possible and from Z = 1 to Z = 0 as late as possible), the only change would be in this case that state 1 will be Z = 1 , instead of Z = 0 .

Another example will be the following flow table and Z matrix.

①	2	④	6
3	②	-	⑥
③	5	-	6
1	⑤	4	⑦

10		01	
	11		11
00			
	10		00

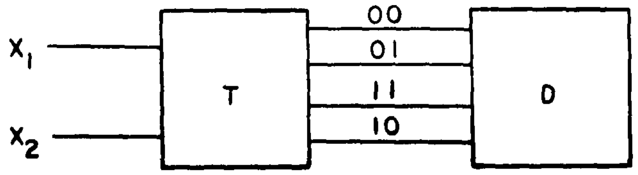
The unstable state 2 is a transient state from ① and ④ to ②. Considering the change from ① to ②, Z changes from 10 to 11 hence Z_1 is to remain 1 and Z_2 may be either 0 or 1 . But from the change from ④ to ② we conclude that Z_2 does not change and Z_1 may be either 0 or 1 , therefore we must assign to entry 2 the value Z = 11 . Entry 3 may be any value because from both ⑥ and

②to ③ Z changes from 11 to 00 . Entry 1 must be 10 and entry 5 may be 00 or 10 . In the same manner we assign values to all the entries. Whenever a choice exists we use the assignment that will best simplify the circuit.

The following circuit is the result after all unspecified entries where assigned values in order to simplify the circuit.

10	11	01	11
00	11	01	11
00	10	01	00
10	10	01	00

A special case is the following network:



T - circuit to be designed
D - Detector

D is a detector that is sensitive to each change in the output. Let the output 01 operate an alarm in D. In such a case while designing the output matrix it is not allowed to let the unstable state 2 [between stable states ①= 00 and 2 = 11] to be 01 or 10 if 10 operates another thing in D . In such cases 2 can be 00 or 11 . In any case where the outputs of T lead to another circuit D - the inputs of which are the outputs of T the designer has to be very careful when assigning the unstable states of the output matrix, and to consider the effects of each assignment on the operation of D . In circuits where the outputs are independent (i.e. each output controls its own operation, as green and red lights) the technique is as described in the preceding pages. It is worthwhile to note that the design of the output matrix is some times easier by using the methods of Huffman and Moore rather than the method of Mealy. While Huffman and Moore associate the outputs with the states of the machine, Mealy associates the outputs with the inputs to the machine in each of its states. While the method of Mealy is recommended for synchronous circuits it is not advantageous for asynchronous circuits.

III. MINIMIZING OF INCOMPLETELY SPECIFIED SEQUENTIAL SWITCHING CIRCUITS

A. INTRODUCTION

Given a flow table of a sequential switching function in which some of the entries are not specified, Paull and Unger[10] in their paper developed a method of finding the maximal compatibles, in order to minimize the number of states in the flow table.

A brief review of some of the theorems and definitions in the above mentioned paper will be given here, as this work is based on them.

A grouping of all the rows of a flow table into C sets will be said to be closed, if every set implied by any C_i is included in one of the C sets.

" A compatible is a set of states of a flow table that constitutes one member of some closed grouping" , i. e. states that do not contradict in their outputs may form a compatible if their input entries belong to another set of states in the flow table.

A set of states is a compatible if, and only if every pair of states in that set is a compatible (i. e. (1, 2, 3) are compatible only if (1, 2) (2, 3) (1, 3) are compatibles).

A maximal compatible (MC) is a compatible that is not included in any larger compatible.

According to the method of Paull and Unger one must find the MC' s in order to reduce the number of rows in the flow table. However the main goal is to achieve the minimal closed set of compatibles that will give the solution. The process developed in this paper results in:

1. Eliminating the "Implication Table" in the form given in reference 10.

2. Eliminating the need to find all MCs.

3. Simplifying the technique of finding the MCs.

This method can as well be applied to completely specified sequential switching functions.

B. MAXIMAL COMPATIBLES - PROCESS FOR FINDING THEM

The method will be illustrated by several examples (some of which are taken from the above mentioned paper) and the following rules:

1. Draw an array of points, each of which represents a row of the flow table. The points are numbered to correspond with the numbers of the states of the flow table.

2. Check each pair of states in the flow table for compatibility. If the pair is a compatible set of states draw a line between the correspondingly numbered points.

3. If for a pair of states the outputs are not contradictory, but the entries in the flow table are not the same - draw an interrupted line between the correspondingly numbered points and write the conditions for compatibility in the space.

4. If the outputs are contradictory do not draw a line.

5. After checking all possible pairs, check if there is any contradiction to the conditions for compatibility as found according to rule 3. If there is a contradiction (i. e., if the condition is ab but between points ab there is no line drawn), cross the condition and ignore the interrupted line. If there are no contradictions to the conditions the interrupted lines have the same significance as the solid lines.

The exact method to check states for compatibility was described in the previous chapter. According to theorem 2 in the above mentioned paper we get the following rule for finding directly the maximal compatibles:

6. In the graph developed according to rules 1-5 check for complete polygons. A complete polygon is one, in which all the possible diagonals exist (i. e., $(n-3)\frac{n}{2}$ diagonals, n = number of sides in the polygon).

The states which correspond with the numbered points of the complete polygon are compatible. Start first from the highest order polygon and proceed checking through lower orders down to triangles and simple lines.

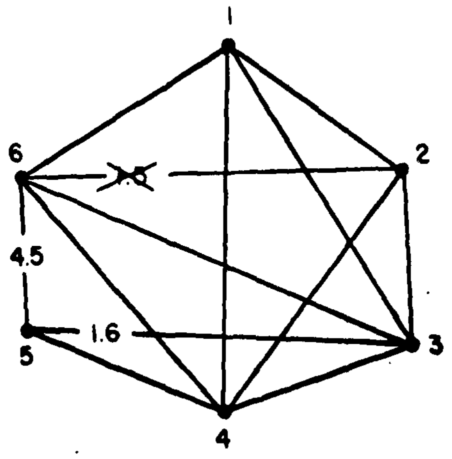
After a complete polygon of the order n has been found all polygons of the order n-1, the verticles of which are included in the polygon of order n do not need further attention. All possible complete polygons which are not included in higher order polygons give the maximal compatibles.

In the flow table P. S. means present state. N. S. - next state. x_1 x_2 ... are always the imputs.

<u>Ex. 3.1</u>

P. S.	N. S. X_1	N. S. X_2	OUTPUTS X_1	OUTPUTS X_2
1	-	-	0	-
2	-	1	0	-
3	1	-	-	-
4	-	-	-	0
5	6	4	1	-
6	-	5	-	-

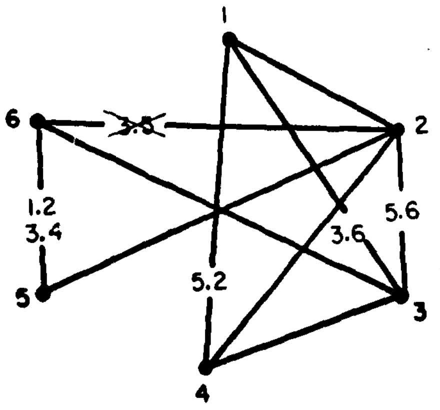
There are 3 complete squares .
(1, 2, 3, 4,) (1, 3, 4, 6,) (3, 4, 5, 6,)



<u>Ex. 3.2.</u>

P. S.	N. S. X_1	X_2	X_3	X_4	OUTPUTS X_1	X_2	X_3	X_4
1	-	3	5	2	-	1	1	1
2	5	-	-	-	0	-	-	-
3	6	6	-	-	0	1	-	-
4	-	-	2	-	-	-	1	-
5	-	6	1	4	-	0	0	1
6	3	-	2	3	0	-	0	1

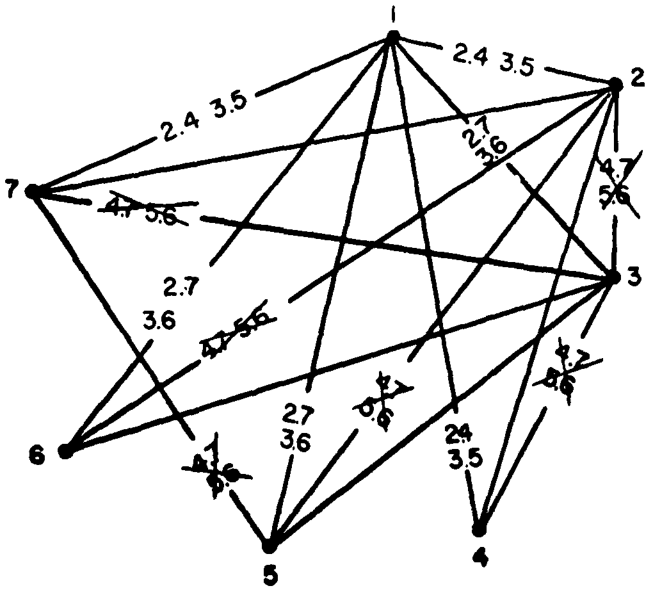
(1, 2, 3, 4) (5, 6) (3, 6) (2, 5)

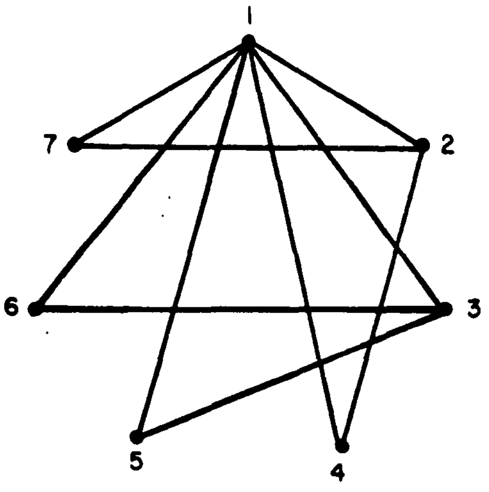


<u>Ex. 3.3.</u>

P. S.	N. S. X_1	N. S. X_2	OUTPUT X_1	OUTPUT X_2
1	2	3	-	-
2	4	5	-	-
3	7	6	-	-
4	4	5	0	1
5	7	6	1	1
6	7	6	1	0
7	4	5	1	1

As there are many "crossed" lines in this example it is better to redraw the graph without the "crossed" lines, in order to make it clearer.





The MCs are: (1, 2, 4) (1, 2, 7) (1, 3, 6) (1, 3, 5) .

But with some experience one can see that in column X_1 the entries 4 and 7 are used many times. It is advisable, therefore, to start checking the states 4 and 7 which contradict in their outputs. Hence whenever the condition 4, 7 exits - the line is eliminated. Same for entries 5 and 6 under column X_2 .

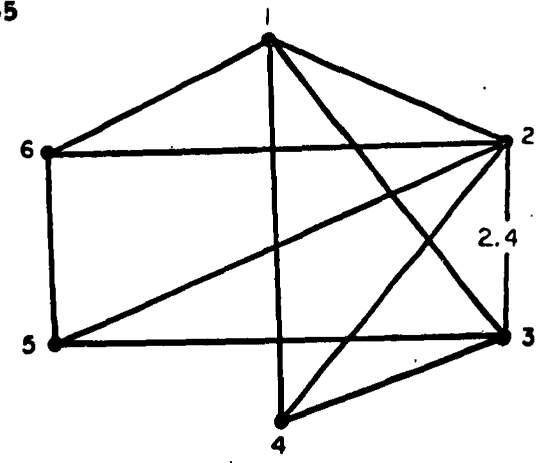
This checking saves time and simplifies the graph.

C. THE TECHNIQUE OF SIMPLIFYING FLOW TABLES

Another advantage of this technique is that it simplifies the method of finding the lower bound of the MCs, without finding all the MCs in advance.

7. A simpler solution is achieved if every point of the graph appears only once in the chosen compatibles, and when the least number of closed sets of compatibles is used.

According to rule 7 we have two criteria, which determine the minimality between the reduced flow tables. First - the minimal flow-table is that flow table which contains the least number of states (which means the least number of compatibles). Second - the minimal expression for the states will be achieved when the flow-table has the maximum number of dashes ("don't care" terms). Hence, when every point will appear only once in the compatibles. After finding the simplest solution it is necessary to check if the set of compatibles is closed. If the set that includes each point only once is not closed (i.e., the reduced flow-table cannot be constructed with the lowest bound), we must expand some compatibles or add more compatibles.



Ex. 3. 4.

P. S.	N. S. X_1	X_2	OUTPUTS X_1	X_2
1	-	-	0	-
2	2	-	-	-
3	4	-	-	0
4	-	4	0	0
5	-	5	1	-
6	5	6	-	1

(1, 2, 3, 4) (2, 3, 5) (2, 5, 6) (1, 2, 6) - Maximal Compatibles
(1, 2, 3, 4) (5, 6) or (1, 3, 4)(2, 5, 6) - are the solution.

In example 3. 4 - 4MCs have been found from the graph. The lower bound, however, is 2 MCs (1, 2, 3, 4) (2, 5, 6).

But we can satisfy rule 7 by the two sets (1, 2, 3, 4) (5, 6) or (1, 3, 4) (2, 3, 6), which cover all points of the graph and form a closed set, and therefore make it unnecessary to find the rest of the possible compatibles.

The reduced flow tables are:

(1, 2, 3, 4) = A (5, 6) = B

P. S.	N. S.	OUTPUT
A	A A	0 0
B	B B	1 1

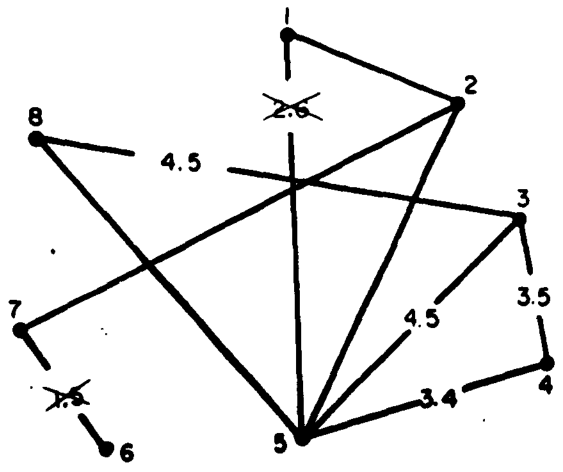
(1, 3, 4) = C (2, 5, 6) = D

P. S.	N. S.	OUT PUT
C	C C	0 0
D	D D	1 1

The flow-tables are exactly the same.

Ex. 3. 5.

Inp.	00	01	11	10	00	01	11	10
P. S.		N. S.				OUT		
1	1	-	6	-	0	-	0	-
2	2	-	-	2	0	-	-	1
3	3	5	2	-	1	1	0	-
4	4	3	7	1	1	1	0	0
5	-	4	2	-	-	1	0	-
6	3	-	8	5	0	-	1	1
7	-	7	5	1	-	0	1	1
8	8	4	2	6	1	1	0	1



The MCs are: (3, 4, 5) (1, 2) (2, 7) (2, 5) (6) (3, 5, 8)

However, according to rule 7 we can conclude what the lower bound will be without knowing the MC' s.

From the graph we see that point 1 is included only in the compatible (1, 2), hence we need this set. In the same manner we find that all points are included in the following compatibles: (1, 2) (3, 4, 5) (6) (7) (8)
Another possibility is: (1, 2) (3, 5, 8) (4) (6) (7)

Checking if the first set is closed we find that (3, 4, 5) implies (2, 7). A check between points 2, 7 shows that a line exists. Hence we may expand (2) into the compatible (2, 7). The set (2, 7) does not imply any new set, hence the solution is:

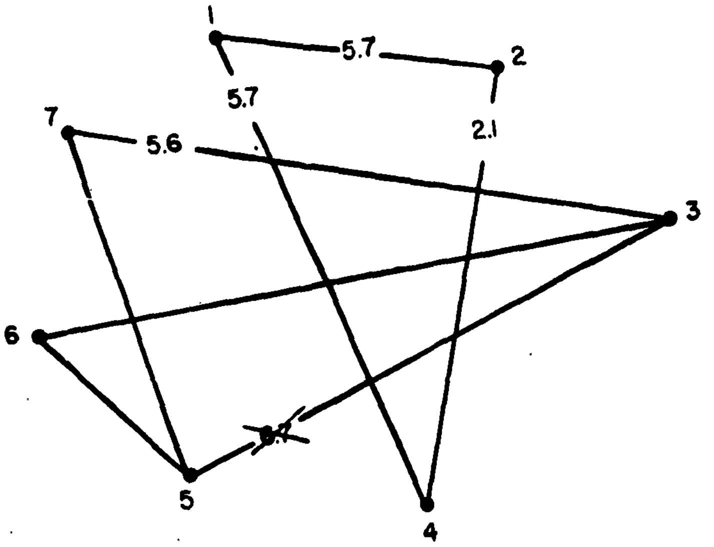
(1, 2) (3, 4, 5) (6) (2, 7) (8) .

In the same manner we find that for the second possibility the result is:

(1, 2) (3, 5, 8) (4, 5) (6) (7)
but (4, 5) implies (3, 4) (2, 7), therefore the first solution was a better one.

Ex. 3. 6.

P. S.	N. S.		OUTPUT	
	X_1	X_2	X_1	X_2
1	5	-	1	0
2	7	2	-	0
3	6	3	-	1
4	7	1	1	0
5	7	-	-	1
6	-	-	0	1
7	5	-	1	1



According to rule 7 the sets of compatibles that cover all points are:

1. (1, 2, 4) (3, 7) (5, 6)
2. (1, 2, 4) (5, 7) (3, 6)

A check for closure in set No. 1 will show that (1, 2, 4) implies (5, 7), therefore we have to add this compatible (a line drawn between points 5-7). Set No. 2 is closed - therefore it represents the best solution. The flow-table assumes the following form:

P. S.	N. S.		OUTPUT	
	X_1	X_2	X_1	X_2
A = (1, 2, 4)	B	A	1	0
B = (5, 7)	B	-	1	1
C = (3, 6)	C	C	0	1

Ex. 3. 7 Given the following flow table:

P. S.	N. S.				OUTPUT			
	00	01	11	10	00	01	11	10
1	1	5	-	1	00	01	-	01
2	-	3	2	4	-	10	00	11
3	1	3	-	-	00	10	-	-
4	1	-	-	4	00	-	-	11
5	-	5	6	-	-	01	00	-
6	-	7	6	7	-	10	00	11
7	1	-	-	7	00	-	-	11

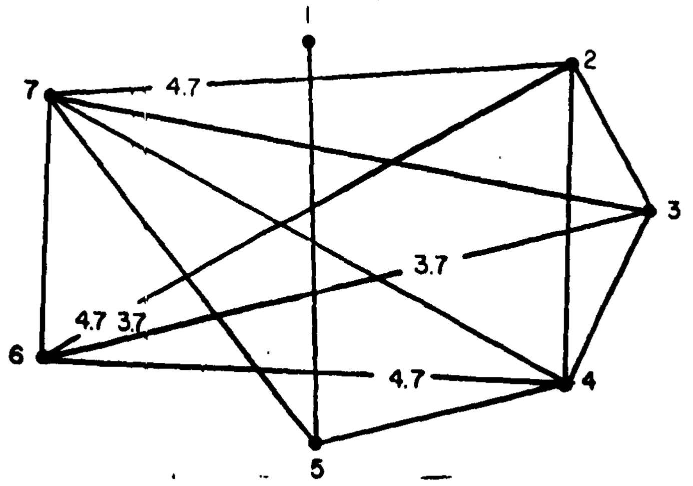
According to the rules of Mealy [3], states the outputs of which are not contradictory are candidates for equivalency. This rule can be applied to the sets: (1, 5), (4, 5, 7), (2, 3, 6) .
(2, 3, 6) implies (3, 7), instead of adding (3, 7) we subdivide (2, 3, 6) into (2, 3) (6).
The closed set is therefore:

A = (1, 5) B = (4, 7) C = (2, 3) D = (6)

P. S.	N. S.				OUTPUT			
	00	01	11	10	00	01	11	10
A	A	A	D	A	00	01	00	01
B	A	-	-	B	00	01	00	11
C	A	C	C	B	00	10	00	11
D	-	B	D	B	-	10	00	11

This flow table cannot be reduced any more, it requires 2 memory elements.

Applying the method developed in this paper results in:



The set (2, 3, 4, 6, 7) (1, 5) satisfies rule 7 and the circuit requires only one memory element.

P. S.	N. S.				OUTPUT			
	00	01	11	10	00	01	11	10
A	B	A	A	A	00	10	00	11
B	B	B	A	B	00	01	00	01

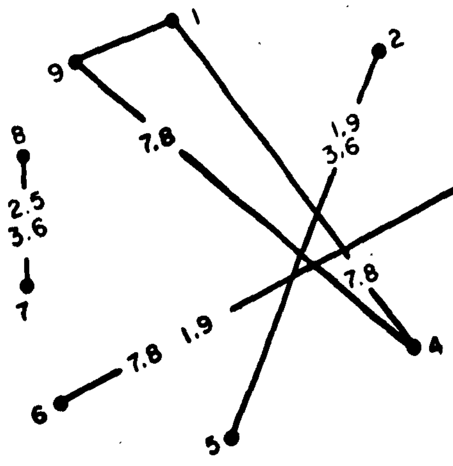
D. COMPLETELY SPECIFIED FLOW TABLES

In the last six examples we have emphasised the method concerning incompletely specified flow-tables. Example 3.8 shows that this method also holds for completely specified flow-tables.

In this type of problems a line can be drawn between two points only if the outputs are the same for the two corresponding states, therefore we have to check only states, the outputs of which are equal.

Ex. 3.8.

P. S.	N. S.		OUTPUT	
	X_1	X_2	X_1	X_2
1	8	1	0	1
2	3	1	1	0
3	8	1	1	1
4	7	4	0	1
5	6	9	1	0
6	7	9	1	1
7	3	2	0	0
8	6	5	0	0
9	8	9	0	1



The solution is given by (1, 4, 9) (2, 5) (3, 6) (7, 8).

E. CONCLUSION

In the preceeding pages a method was developed for simplifying flow tables of completely and incompletely specified sequential switching circuits. The main advantages of this method are that it eliminates the need to find all the MCs of the flow table in order to get a simple solution. It also simplifies the technique of finding the MCs.

The incompletely specified switching circuits are of a nature that allows no unique solution. Therefore checking several possibilities is necessary, but with some experience it is not very difficult, even for larger flow-tables. When the problem is very complicated it is advantageous to find all the MCs first and afterwards apply rule No. 7.

This method, however, does not offer solution to the problem of finding the closed set of compatibles in a more general technique and without the need for checking several possibilities, which sometimes, in larger flow-tables, may be long and tiring.

F. THE CLOSED SET OF COMPATIBLES

The methods described in the previous chapters provided us with a procedure for reducing the number of states in the flow tables and receiving a set of MC' s .

The problem that we face now is how to choose the minimal closed set of compatibles that will cover all states of the machine. In general we dont have any rules to determine which MC' s or compatibles belong to the minimal closed set. It is a mistake to assume that the minimal closed set of compatibles will include any of the MC' s at all.

For every flow table the number of possibilities to find closed sets of compatibles varies and there are no techniques or rules according to which one can decide what is the number of possibilities and hence it is impossible to find the set of all closed sets of compatibles. In simple cases (ex. 3. 9 - 3. 12) after several tests one can find the minimal set (which is at least the lower bound of MC' s).

In large and complicated flow tables the number of the closed sets in large, and up to now there are no methods to determine how to test all sets for closure and minimality, and we dont have any previous knowledge about the minimal number of compatibles, (except of the fact that it cannot be less than the lower bound of the MC' s).

In the following pages a technique is presented for the determination of the minimal closed set of compatibles. However, this technique is not complete and when the flow tables are large and complicated the tests needed according to the following technique are complicated and very tireing.

This technique uses the table used by McCluskey in his paper "Minimization of Boolean Functions" (Published in BSTJ in November 1956), McCluskey called this table "Prime Implicant Table". In this chapter we shall use the name "MC' s table".

In order to find the minimal closed set of compatibles one must follow the following steps:

Step No. 1. Given the MC' s, construct the MC' s table which shows the relations between the MC' s and the states of the flow table.

Step No. 2. States that are included only in one compatible are circled and the compatible is checked by this sign (✓). States that are circled are also checked by the same sign.

Step No. 3. Try to expand the checked compatibles from a single state up to the point where it implies other compatibles. Check the corresponding states.

Step No. 4. States that are not checked are tested for inclusion in the compatibles already checked. The last step requires many experiments, tests for closure etc.

The main idea in the following method is that we start with only the circled states assuming that each will form a compatible by itself. Only afterwards we expand this compatible in such a manner that the additional states will not imply any new unnecessary compatibles or the compatibles which are implied by the expansion of the circled states introduce only new states which are not yet included in any compatible. The technique will best be illustrated by the following examples:

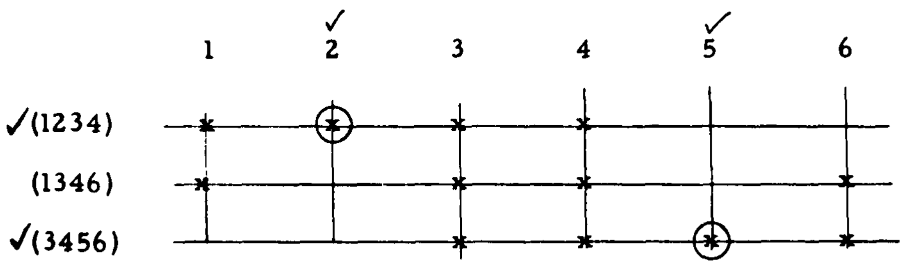
Ex. 3.9

The following flow table and the corresponding MC' s are given. Find the minimal closed set of compatibles.

P. S.	I_1	I_2
1	- . 0	- . -
2	- . 0	1 . -
3	1 . -	- . -
4	- . -	- . 0
5	6 . 1	4 . -
6	- . -	5 . -

(1346) (1234) (3456)

The MC' s table is



The number of rows in the MC' s table is equal to the number of MC' s. The numbers of columns corresponds to the numbers of the states in the flow table. The X' s in column 1 indicate that state No. 1 is included in both compatibles (1234) and (1346). The circled states 2 and 5 indicate that each of the states is included in only one compatible.

In this case we checked the compatibles (1234) and (3456), this mark does not imply that in the final solution these MC' s must appear, it only indicates that subsets of these MC' s must be included in the final solution. The subsets must include only states 2 and 5 but they may include any other states which belong to the same MC .

However, in this case the checked MC' s constitute the lower bound of MC' s hence we try to expand the compatibles 2 and 5 according to Step No. 3.

To state 5 we can add states 4 and 6 and the compatible (456) does not imply any other compatible. Note that if we would have included state 3 in this compatible it would have implied (1, 6). To state 2 in the same manner we can add states 1 & 3 and the final solution is:

(1, 2, 3) (4, 5, 6) .

Ex. 3.10

P. S.	I_1	I_2
1	5 . 1	- . 0
2	7 . -	2 . 0
3	6 . -	3 . 1
4	7 . 1	1 . 0
5	7 . -	- . 1
6	- . 0	- . 1
7	5 . 1	- . 1

The MC' s are:

(1, 2, 4) (3, 7) (5, 6) (5, 7) (3, 6)

The MC' s table is:

	✓ 1	✓ 2	✓ 3	✓ 4	✓ 5	✓ 6	✓ 7
✓(1, 2, 4)	⊗	⊗		⊗			
(3, 7)			x				x
(5, 6,)					x	x	
✓ (3, 6)			x			x	
✓(5, 7)					x		x

Circle 1, 2 and 4.

This circle does not necessarily imply that 1, 2 and 4 will be included in the same MC, we may decide to take each state or combination of states separately if the MC (1, 2 ,4) implies other MC's which are not necessary.

However the first logical test is to see whether it is possible to take the MC (1,2, 4). (1, 2 4) implies (5, 7). But as (5, 7) does not imply any other compatible and it includes only "new" states we decide to include (5, 7) in the minimal closed set. The only states that are not checked yet are 3 and 6 and they form a MC.

The final solution is:

(1, 2, 4) (5, 7) (3, 6)

This solution is the lower bound, hence we are sure it is the minimal solution.

Ex. 3.11

Given the flow table of Ex. 3.5 the corresponding MC' s table is:

	1	2	3	4	5	6	7	8
	✓	✓		✓		✓	✓	✓
(3, 4, 5)			x	⊗	x			
(1, 2)	⊗	x						
(2, 7)		x					⊗	
(2, 5)		x			x			
(6)						⊗		
(3, 5, 8)			x		x			⊗

Circle 1, 4, 6, 7, 8. State 2 is already included in the compatibles (1, 2) and (2, 7) a test shows that (2, 7) implies (1, 2) and (1, 2) does not imply anything hence we take the MC (1, 2).

This choice now depends upon the rest of the MC' s (i. e., if the other MC' s do not imply (2, 7)).

The only states which are not yet checked are 3 and 5. There are several possibilities to include 3 & 5 in the compatibles which are already marked.

1) (3, 4, 5)
2) (3, 5, 8)
3) (3, 4) (5)
4) (3) (4, 5)
5) (3, 5) (4)
6) (3) (4) (5)

(3, 4, 5) implies (2, 7) hence we must expand (7) into (2, 7) and state 2 appears twice (2, 7) and (1, 2). All other possibilities are more complicated.

The final result is:

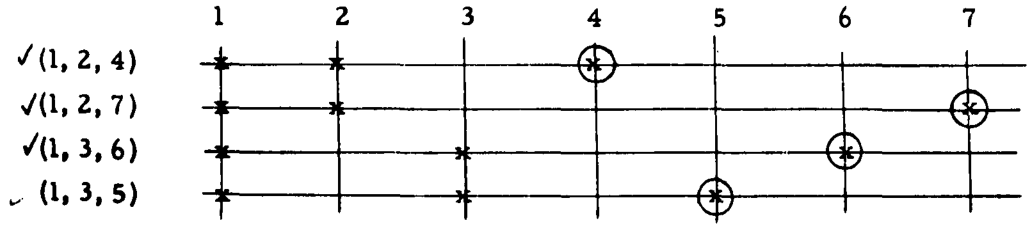
(3, 4, 5) (1, 2) (2, 7) (6) (8)

Ex. 3.12

P. S.	I_1	I_2
1	2 . -	3 . -
2	4 . -	5 . -
3	7 . -	6 . -
4	4 . 0	5 . 1
5	7 . 1	6 . 1
6	7 . 1	6 . 0
7	4 . 1	5 . 1

The MC's are:

(1, 2, 4) (1, 2, 7) (1, 3, 6) (1, 3, 5)



First solution is the four MC's. However, even though the number of compatibles cannot be reduced, the flow table can be simplified by adding more don't care conditions.

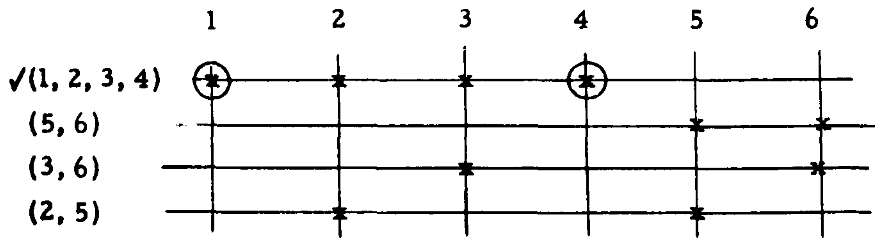
State 1, 2, 3 are not circled, we add 1, 2 to state 4 and it implies (3, 5), hence the final solution is:

(1, 2, 4) (3, 5) (6) (7)

Ex. 3.13

P. S.	I_1	I_2	I_3	I_4
1	- . -	3 . 1	5 . 1	2 . 1
2	5 . 0	- . -	- . -	- . -
3	6 . 0	6 . 1	- . -	- . -
4	- . -	- . -	2 . 1	- . -
5	- . -	6 . 0	1 . 0	4 . 1
6	3 . 0	- . -	2 . 0	3 . 1

The MC's are: (2, 5) (1, 2, 3, 4) (3, 6) (5, 6)



Circle 1 & 4. A test for (1, 2, 3, 4) show that it implies all other compatibles. However (1, 4) implies (2, 5) which does not imply any new compatible. (3, 6) is added and the final result is:

(1, 4) (2, 5) (3, 6)

Ex. 3.14

P. S.	I_1	I_2	I_3	I_4
1	9 . 9	5 . 0	8 . 0	- . -
2	9 . 0	5 . -	5 . -	2 . 1
3	7 . 0	5 . -	8 . -	3 . 0
4	7 . -	6 . -	- . 0	2 . 1
5	7 . 0	- . 0	- . -	- . -
6	3 . -	- . 0	3 . -	3 . 0
7	1 . -	7 . 0	3 . 0	- . -
8	2 . 0	9 . 0	- . 0	- . -
9	1 . 0	4 . 0	3 . 0	- . -

The MC' s for this flow table are:

(1, 2, 4, 5, 7, 8) (1, 2, 5, 7, 9) (1, 3, 5, 6, 7, 9) (1, 3, 5, 7, 8)

The MC' s table is:

				✓		✓			
	1	2	3	4	5	6	7	8	9
✓ (1, 2, 4, 5, 7, 8)	x	x		x	x		x	x	
(1, 2, 5, 7, 9)	x	x			x		x		x
(1, 3, 5, 6, 7, 9)	x		x		x	x	x		x
(1, 3, 5, 7, 8)	x		x		x		x	x	

In order to test the compatibles for closure we can start with either of the circled states. Starting with state 6 we try to add to it the states 1, 3, 5, 7, 9 in such a manner that it will imply the least number of other compatibles.

Adding 3, 5 to 6 we get (3, 5, 6) which implies (3, 8) and (3, 7). (3, 7) implies (1, 7) which results in the MC (1, 3, 5, 6, 7, 9). This MC implies itself and the compatibles (3, 8) (4, 5, 7). (3, 8) implies (2, 7) and (5, 9(hence we must expand (4, 5, 7) into (2, 4, 5, 7) which is a compatible also.

Therefore the final solution is:

(1, 3, 5, 6, 7, 9) (3, 8) (2, 4, 5, 7)

The states that appear twice in these compatibles are: 3, 5, 7 , and it can be easily shown that there is no way to simplify this solution.

Concluding Remarks

The method presented in Chapter III simplifies the procedure of reducing the number of states in flow tables which are not completely specified. The reduction method consists of two parts, finding the MC' s and from which deteriming the minimal closed set of compatibles that will form the reduced flow table.

In Chapter III (A - D) attempt has been done to combine these two parts and to find directly the minimal closed set of compatibles. In several examples this technique succeeded. However no general method has been found yet, and the reduction problem of incompletely specified flow tables requires trials of the sort of "cut and try" .

IV. SYNCHRONOUS CIRCUITS

INTRODUCTION

The synchronous circuits differ from the asynchronous circuits in that they are driven by a clock pulse which controls the time at which the circuit may change state. In the synchronous circuits electronic devices are used in order to realize the alegebraic equations. Instead of levels of voltage which were used in the asynchronous circuits to represent the variables, in the synchronous circuits the variables are represented by pulses.

There are several types of pulse circuits, however the most important circuit used in the synchronous circuits is the Flip-Flop.

Due to the fact that all transitions from states are controlled by a clock pulse, the races which were a very important problem in the asynchronous circuits do not occur in the synchronous circuuts, because the different speed of operating and releasing of the electronic devices does not effect the final state.

The main tool in solving the design problems of the synchronous circuits is the state diagram, described in Chapter II B and developed by Moore[4] and Mealy[3].

The state diagram is constructed by examining each state and determining the next state for each of the conditions of the inputs.

After a state diagram is constructed it is simplified (an intermediate step of constructing the state table can be used here), and a flip-flop state is assigned to each of the system states.

In the synchronous circuits all inputs and outputs are pulses, the shapes and durations of which are assumed to be compatible with the electronic devices.

The output pulse is obtained if and only if pulses occur at the inputs in a certain specified order.

The Design of the Synchronous Machine

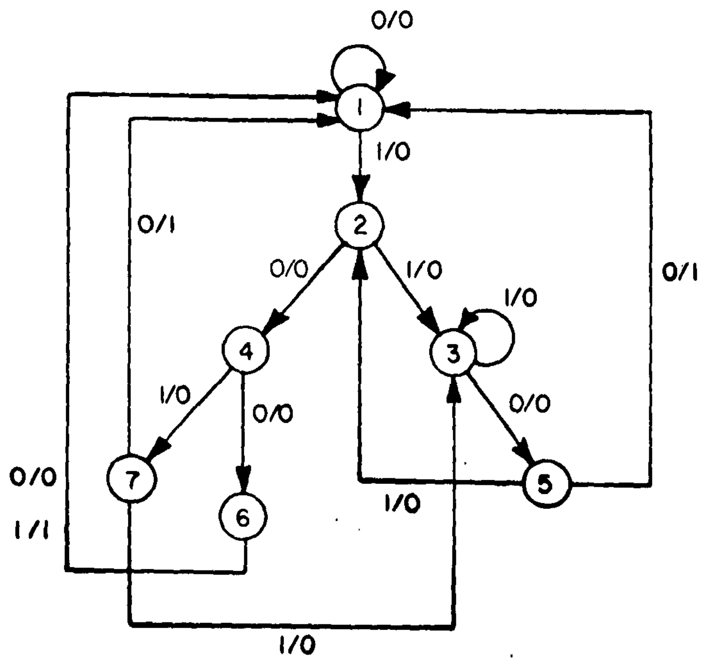
The design of the synchronous sequential machine will be illustrated by the following examples:

Ex. 4.1

Design a circuit that will give an output whenever one of the following sequences of pulses occur:

1100, 1010, 1001.

Assume that the pulse train is not interrupted and that after an output is given, the machine starts from the initial state. According to Mealy's method the following state diagram is constructed.



State 1 is the initial state, a zero input does not start any sequence, hence the machine remains in the same state.

State 2 is the state that carries a "one" .

A zero can start either sequence 1010 or 1001, a one can start 1100. In state 3 any change of input that is not zero does not change the state of the machine (i. e., 1111 can be the start of the sequence 1100) therefore it waits for a "zero" pulse, which moves the machine to state 5. Another zero in state 5 results in an output as the first sequence is complete. However, a one in state 5 means a sequence like 1101 which cannot form any of the other sequences (which result in an output) but it can form the beginning of a sequence, therefore an arrow leads to state 2. The last arrow which may need an explanation is from state 7 to 3.

To state 7 the machine goes when the pulses were 101, an additional 1 will give 1011 which does not result in an output, but can be the beginning of the sequence 1100, which brings the machine to state 3.

The corresponding flow table is:

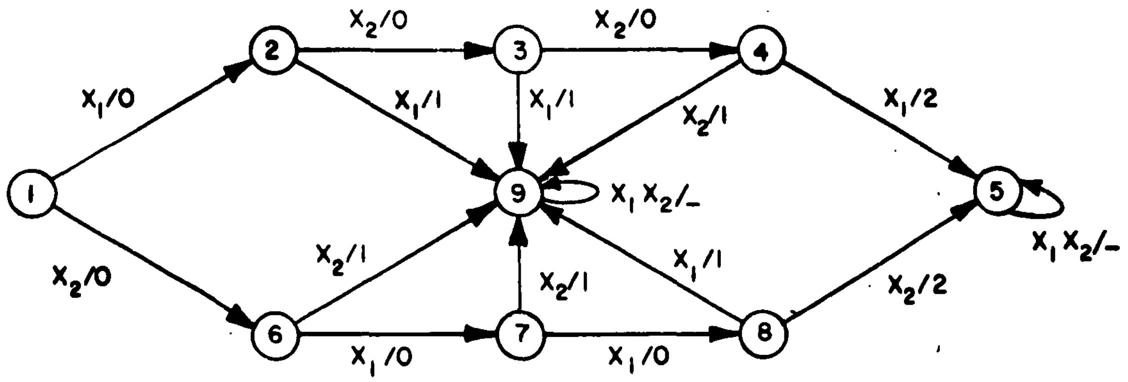
P. S.	N. S. 0	N. S. 1	OUTPUT 0	OUTPUT 1
1	1	2	0	0
2	4	3	0	0
3	5	3	0	0
4	6	7	0	0
5	1	2	1	0
6	1	1	1	0
7	1	3	1	0

This flow table (which is called also state table) cannot be reduced anymore, the realization of the machine requires 3 flip-flops.

Ex. 4.2

A lock of a safe is controlled by pulses from two push-buttons X_1 and X_2. The lock opens if X_1 X_2 are operated (depressed and released) in the sequence X_1 X_2 X_2 X_1 or X_2 X_1 X_1 X_2. The lock opens and remains open after the last pulse in the sequence. If the buttons are pushed in any other sequence, an alarm is stopped, and the lock is closed by an external action.

It is obvious that we need nine states, four states for each sequence and one state for the alarm. The state diagram is as follows:



State 1 is the initial state, state 9 is the alarm state. Output 1 is the alarm, output 2 is the lock of the safe. X_1 moves the machine to state 2, X_2 moves it to state 3 etc., till the sequence X_1 X_2 X_2 X_1 is performed and the safe opens.

Pressing first X_2 bring the machine to state 6.

However if the required sequences are interrupted the machine moves to state 9 and the alarm is on.

The state table corresponding to this state diagram is:

P. S.	N. S.		OUTPUT	
	X_1	X_2	X_1	X_2
1	2	6	0	0
2	9	3	1	0
3	9	4	1	0
4	5	9	2	1
5	5	5	-	-
6	7	9	0	1
7	8	9	0	1
8	9	5	1	2
9	9	9	-	-

In state 5 the lock is open, hence whatever the input is the output is not specified. Same for state 9.

Applying the method for reducing the number of state in state tables we get:

States 5 and 9 are equivalent. A = (5. 9).

Due to this fact the next states of states 4 and 5 are the same, therefore A = (4, 5, 9). The final state table is:

P. S.	N. S.		OUTPUT	
	X_1	X_2	X_1	X_2
1	2	6	0	0
2	A	3	1	0
3	A	A	1	0
A	A	A	2	1
6	7	A	0	1
7	8	A	0	1
8	A	A	1	2

V. SECONDARY STATE ASSIGNMENT

INTRODUCTION

The problem of assigning the binary variable states to represent the internal states of the machine is the most difficult and complicated part in the whole design of the sequential machine.

A poor assignment may result in a circuit that will be very complicated and will require more secondary elements than are needed with a better assignment.

Let n be the number of rows in the reduced flow table. Then the minimal number S of the secondary elements is:

$$S \geq \log_2 n$$

In other words, S secondary elements have 2^S secondary states which may be assigned to a flow-table of 2^S rows. But as will be seen in this chapter, this lower bound of the number of memory elements cannot always be achieved.

There are two criteria to check whether the assignment is optional. First, if the number of secondary elements is the minimal possible to realize the flow-table. Second, if the circuit that was designed is the least complicated, i.e., requires minimum number of logic devices.

An example taken from reference 14 will briefly illustrate the importance of the second criteria.

Ex. 5.1

Given the following flow table:

	INPUTS		
P. S.	0	1	Z
0	3	2	0
1	5	2	0
2	4	1	0
3	1	4	1
4	0	3	0
5	2	3	0

We shall consider the following assignments and compare the equations for the sequential machine.

		Y_1	Y_2	Y_2
0	$\rightarrow$	0	0	0
1	$\rightarrow$	0	0	1
2	$\rightarrow$	0	1	0
3	$\rightarrow$	0	1	1
4	$\rightarrow$	1	0	0
5	$\rightarrow$	1	0	1

assignment a.

		Y_1	Y_2	Y_3
0	$\rightarrow$	1	1	0
1	$\rightarrow$	1	0	1
2	$\rightarrow$	1	0	0
3	$\rightarrow$	0	0	0
4	$\rightarrow$	0	0	1
5	$\rightarrow$	0	1	0

assignment b.

The exitation matrices for these assignments are:

$Y_1\ Y_2\ Y_3$ \ x	0			1			Z
0 0 0	0	1	1	0	1	0	0
0 0 1	1	0	1	0	1	0	0
0 1 0	1	0	0	0	0	1	0
0 1 1	0	0	1	1	0	0	1
1 0 0	0	0	0	0	1	1	0
1 0 1	0	1	0	0	1	1	0

assignment a.

$Y_1\ Y_2\ Y_3$ \ x	0			1			Z
1 1 0	0	0	0	1	0	0	0
1 0 1	0	1	0	1	0	0	0
1 0 0	0	0	1	1	0	1	0
0 0 0	1	0	1	0	0	1	1
0 0 1	1	1	0	0	0	0	0
0 1 0	1	0	0	0	0	0	0

assignment b.

The corresponding equations for the secondary variables are:

assignment a.

$$\begin{cases} Y_1 = \bar{y}_1 \bar{y}_2 y_3 \bar{X} + y_2 \bar{y}_3 \bar{X} + y_2 y_3 X \\ Y_2 = \bar{y}_2 X + \bar{y}_1 \bar{y}_2 \bar{y}_3 + y_1 y_3 \\ Y_3 = y_1 X + y_2 \bar{y}_3 X + y_2 y_3 \bar{X} + \bar{y}_1 \bar{y}_2 \bar{X} \end{cases}$$

assignment b. $$\begin{cases} Y_1 = y_1 X + \bar{y}_1 \bar{X} \\ Y_2 = y_3 \bar{X} \\ Y_3 = \bar{y}_2 \bar{y}_3 \end{cases}$$

From the two sets of equations we can see that the secondary assignment is of great significance. The question is how to assign 2^S secondary states to the n rows of the flow table.

In the asynchronous machines the secondary assignment has to be such that the races will be avoided.

In the synchronous machines the problem is to assign the binary flip-flop configuration to each of the states of the flow table, and to design the exitation of the set and the reset of the flip-flops.

We shall consider both, the synchronous and asynchronous circuits, using alternately the methods of Huffman and Mealy.

In general there is no method to find the best assignment; there are several techniques which do not promise a good result, but are of some help to the designer.

The first problem is to decide what is the number of the essentially different assignments possible for an n - row flow table.

McCluskey and Unger in their paper proved that the number of distinct row assignments for an n - row flow table is:

$$N = \frac{(2^S - 1)!}{(2^S - n)!\, S!}$$

where $n \leq 2^S$

If $n < 2^S$ the total number of possible assignment is $\dfrac{2^S!}{(2^S - n)!}$

However, many of these assignments are essentially equivalent to each other, since they differ only in that some of the states variables have merely been relabelled.

S binary variables have S ! permutations, and there are 2^S ways of complementing them and thus S ! 2^S ways of permuting and complementing the S binary variables.

But in order to get essentially different assignments the number of possibilities decreases by S ! 2^S and it becomes:

$$N = \frac{2^S!}{(2^S - n)!\ S!\ 2^S} = \frac{(2^S - 1)!}{(2^S - n)!\ S!}$$

This formula is restricted to the synchronous functions. In the case of asynchronous functions it is necessary to check each assignment if it does not lead to a critical race condition.

If more than S state variables are used, this formula becomes unsatisfactory since some of the assignments counted are degenerate. For example if $n = 4$ $S = 2$ then $N = 3$.

But if we use $S = 3$ then according to this formula $N = 35$. As stated before this result is not exact because some of the assignments have some state variable constant for all rows, or have two or more identical state variables.

Several other formulas were given for special cases. In the case described before it was proven that $N = 29$ instead of 35.

1) ASYNCHRONOUS CIRCUITS

A. Races

A race occurs when a single change of the internal states involves the change of two or more of the secondary variables. Or when a single change of the input causes a change of two or more of the secondary variables simultaneously. A change of the inputs causes the machine to change an internal state. If the machine can reach its destination by passing through several unstable states, we say that a race exists. However, if the machine does not reach its required destination, and reaches instead another stable state, we say that a critical race exists, and the machine is non-deterministic. Moore[4] calls these kine of machine - not speed independent. In this definition he means that the machine depends on the operating and releasing speed of the secondary elements.

Let us illustrate the problem of races by the following example.

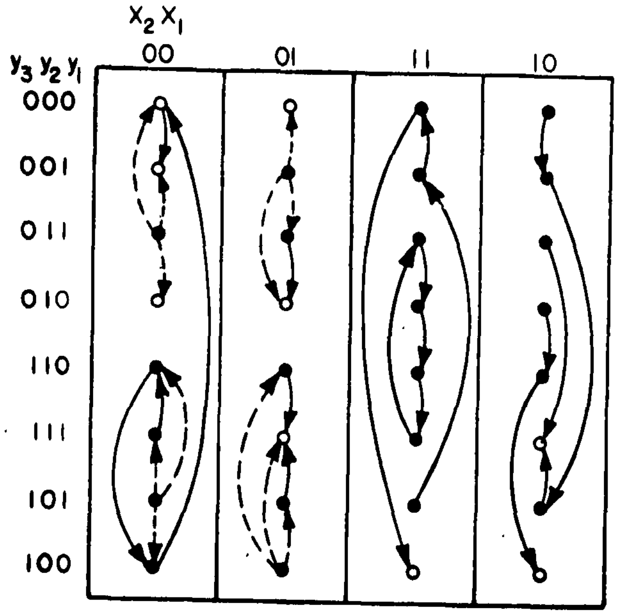
Ex. 5.2

Given the following exitation matrix (a). In fig. (b) a circle describes a stable state, a dot describes an unstable state.

The solid lines indicate transition from unstable state to the corresponding stable states. A race condition is indicated by the dotted lines.

	X_2X_1			
$Y_3Y_2Y_1$	00	01	11	10
0 0 0	0 0 0	0 0 0	1 0 0	0 0 1
0 0 1	0 0 1	0 1 0	0 0 0	1 0 1
0 1 1	0 0 0	0 1 0	0 1 0	1 1 1
0 1 0	0 1 0	0 1 0	1 1 0	1 1 0
1 1 0	1 0 0	1 1 1	1 1 1	1 0 0
1 1 1	1 1 0	1 1 1	0 1 1	1 1 1
1 0 1	1 1 0	1 1 1	0 0 1	1 1 1
1 0 0	0 0 0	1 1 1	1 0 0	1 0 0

(a)



(b)

Each next state will be denoted by the five digits $y_3\ y_2\ y_1\ X_2\ X_1$.

A check through column 00 will show two race conditions. The state 01100 is unstable, and as indicated its desired destination is state 000. This transition requires a change of two variables y_1 and y_2 . Hence this circuit depends on the speed of the secondary elements. If y_1 is faster than y_2 , the transition will be to state 010 which is stable; therefore, the machine will remain in this state and will not reach its destination 000 . Only when the speed of both secondary variables is the same will the machine reach state 000 . But the probability that both have exactly the same speed is very small, and this becomes a critical race. In this case due to different speeds of releasing of the secondary elements, the machine did not reach its destination and ended in a different state.

The square 10100 is unstable and is to be stabilized by a change to state 110 and from which to 100 $\rightarrow$ 000 . If the speed of both relays is different, the transition may be to any of the three unstable states 110, 111, or 100 . However from each of these states the machine will end in reaching state 000 . Hence this race is not critical. Other critical and non-critical races may be found in column 01 . In column 11 there are no races, but there is a closed loop of unstable states. This situation is not allowed because once the machine enters this cycle, it will continue indefinitely unless a change in the input is made to stop it.

In the synthesis of sequential circuits it is allowed to use non-critical races in order to simplify the circuit, but the critical races or the closed loops must be avoided.

Ex. 5.3.

Given the following flow table, let us check it for races.

	N. S.			
P. S.	00	01	11	10
a	a	b	c	d
b	b	b	d	b
c	c	d	c	c
d	a	d	d	d

Assuming that the machine is in state a, any change of the input requires a change to a different next state. For example, if the next input is 11 , the machine is supposed to change to state c, but if both inputs do not occur at exactly the same time, the machine will be in state b or d , depending on the inputs. This again is a critical race and it must be considered while designing the machine. To avoid this kind of races it is sometimes not allowed to change two inputs or more at a time.

But this flow table has another kind of race, due to the fact that in order to realize it with only two secondary variables for some changes in the inputs, the final state of the machine depends on the speed of the secondary variables.

B. Coding and Assignments to Avoid Races

In order to avoid races each transition is accomplished either by a change of a secondary state in which only one secondary variable changer or by a change of secondary state in which a multiple change of secondary variables does not involve critical races.

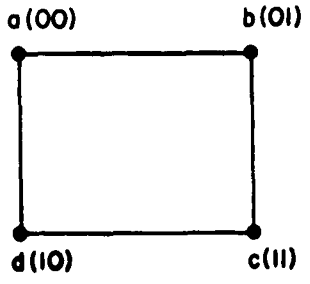
Given a flow table, in order to assign the secondary variables to the rows of the flow table one has to decide which secondary states (rows of the flow table) must be adjacent, (i.e. states which differ in only a single variable).

Caldwell[1] in his book suggested a simple graphical method which - as all graphical methods - has the disadvantage of becoming very complicated as the number of states in the flow table increases. Huffman[2] studied these problems in detail. His methods of coding are based on two standard circuits - the $2S + 1$ realization and the one relay per row realization - . In the first method Huffman proved that $2S + 1$ relays - ($S = \log_2 n$ when n is the number of rows in the flow table) will always avoid any race in any n - row flow table. In the second standard circuit each row of the flow table has an associated relay. However, these methods are not economical; the number of secondary variables which is required is extremely large; and the combinational circuit becomes very complicated.

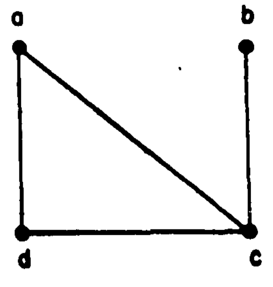
Given a three-row flow table in which it is required that all rows are adjacent to each other. Assigning two secondary variables to these rows may result in critical races. If we assign the secondary states 00 01 11 to the rows of the flow table, the machine may never enter the state 11 if the second relay operates faster, because secondary state 00 will change to 01 as it is a stable state, the machine will remain there even if it was initially intended to reach state 11 . However, if we assign the secondary states 00 01 and 10 to the three rows of the flow table and use the state 11 as a transit unstable state; the machine will operate without critical races. In order to avoid all kinds of races we must use at least 3 secondary variables. It becomes easier to decide which row must be adjacent if we use a simple graphical technique. In the following graphs the vertices are corresponding to the present state of the flow table, and the lines are corresponding to the required transition between the states (or required adjacencies between states).

Ex. 5.4.

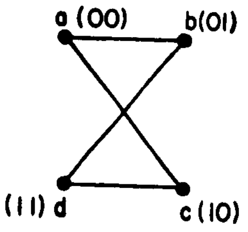
As an example we consider a four-row flow table. We want to determine in what cases can this flow table be realized with 2 secondary variables.



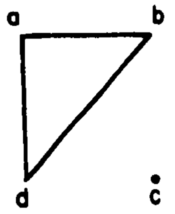
pattern α



pattern β



pattern γ



pattern δ

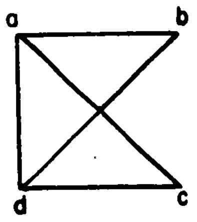
It is obvious that only patterns α and γ can be realized with two secondary variables. We conclude that a pattern can be realized with two secondary variables if it does not contain any triangles and if no vertex is connected to more than two other vertices. Caldwell defines this rule as the "odd-even rule" . According to this rule when two secondary states are adjacent, the code of those states must contain for one state an even number of 1' s and for the other state an odd number of 1' s (00 is considered as even number). In pattern β or δ this rule is violated because there exist in the patterns triangles, and a triangle always violates the odd-even rule. (If a is even, b and d must be odd, but then b and d violate the rule as both are odd).

<u>Ex. 5.5</u>

Given the following flow table (Mealy' s model, outputs omitted).

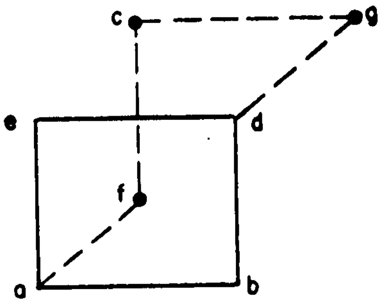
	N. S.			
P. S.	00	01	11	10
a	a	b	d	c
b	a	b	b	b
c	a	c	c	c
d	a	c	d	b

The pattern of this flow table is:



Hense this machine will have at least three secondary variables, and even then non-critical races may still remain in the operation of the machine.

Caldwell suggested a method of "breaking" the triangle i.e., adding another node in such a manner that there will be no triangles in the pattern of the flow table. In our example by adding a node between node a and node d we get:



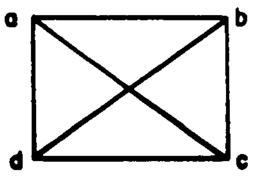
Hence state a is adjacent to states b and e and through the unstable state f, it is "partially adjacent" to state c. Same about states d and c. The dotted lines indicate transition due to non-critical races.

It is worthwhile to mention that this is not a unique solution. In fact the number of possible assignments in this case is very high, and no method exists to determine which solution is better and simpler.

<u>**Ex. 5.6**</u>

Consider the following flow table:

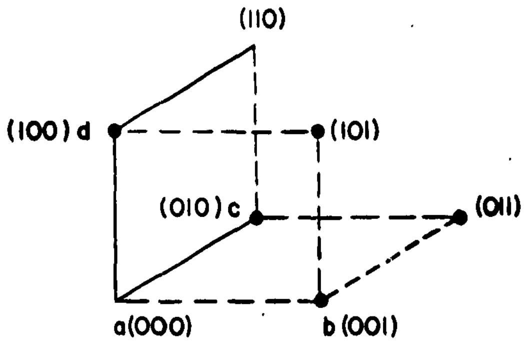
①	3	⑤	⑦
1	④	⑥	7
②	4	5	7
2	③	6	7



pattern for flow table 5.6.

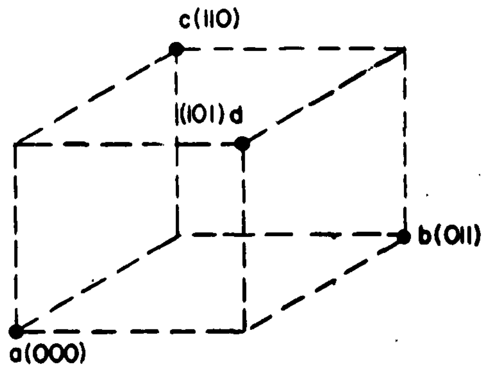
At least 3 secondary variables are needed. In general there are several possible assignments for this flow table.

<u>**Assignment α .**</u>



In this assignment it is only fortuitous that the code of a is 000, and there is no way of knowing if another coding (111 for example) would not result in a simpler combinational solution.

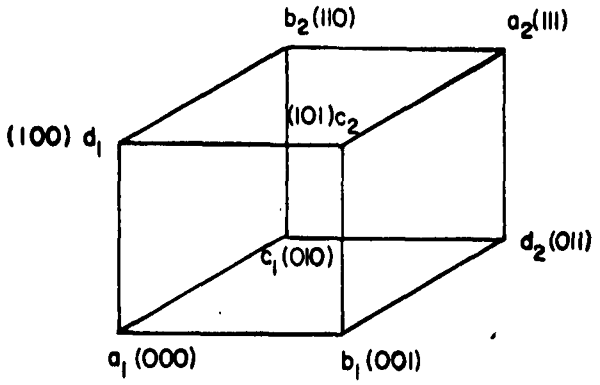
<u>**Assignment β .**</u>



None of the secondary states are adjacent. All transitions between rows require two changes of secondary states.

000	①	3	⑤	⑦
001	1	3	6	7
011	1	④	⑥	7
010	1	4	5	7
110	②	4	5	7
111	2	4	6	7
101	2	③	6	7
100	2	3	5	7

Assignment γ



In this assignment each present state is represented by two different secondary states. All transitions between row-sets require just one time of operate or release. A row-set is any number of secondary states which represents one present state in the flow table. Although the individual states in the matrix are not necessarily adjacent, each pair of row-sets is adjacent to the other three pairs of row-sets.

The output is assigned to both secondary states equally. The non stable states are labeled according to the stable states which they go into.

000	$\enclose{circle}{1_1}$	3_1	$\enclose{circle}{5_1}$	$\enclose{circle}{7_1}$
001	1_1	$\enclose{circle}{4_1}$	$\enclose{circle}{6_1}$	7_1
011	2_1	$\enclose{circle}{3_2}$	6_1	7_2
010	$\enclose{circle}{2_1}$	4_2	5_1	7_1
110	1_2	$\enclose{circle}{4_2}$	$\enclose{circle}{6_2}$	7_2
111	$\enclose{circle}{1_2}$	3_2	$\enclose{circle}{5_2}$	$\enclose{circle}{7_2}$
101	$\enclose{circle}{2_2}$	4_1	5_2	7_2
100	2_2	$\enclose{circle}{3_1}$	6_2	7_1

Flow table for assignment γ.

The row sets can be either as in assignment γ or they can be adjacent, (i.e., a , adjacent to a_2 etc.).

Discussion

The technique which we studied in the last chapter has several limitations.

It is fairly simple for small flow tables and for the cases where no races are allowed (critical and non critical). By constructing the map of Karnaugh we could use this technique for flow tables of at most 16 rows. If non critical races are allowed the number of possible assignments and adjacencies is very large, and the best that can be done is to try several possibilities and to find which will give a simpler solution. However, as the number of possibilities is very large, the probability of picking the best assignment is very small. This technique does not consider the factor of the combinational circuit which is realized from the flow table; the importance of this factor was illustrated in example 5.1.

Finally we can conclude that this technique can be a way to check if there is a possibility to make the assignment with a certain number of secondary variables, and to decide what this minimal nunber is. But this is not a method to determine how to make the coding of the present states nor how many secondary variables to use, (which must be equal to or greater than the number that was found by this technique).

Other techniques, as the relay per row realization, are in general solutions to solve the problem of races, but they require a large amount of secondary variables and logical devices so that it makes this techniques impractical and purely theoretical.

2) SYNCHRONOUS CIRCUITS

C. Secondary Assignments using Partitions.

In the introduction to this chapter it was mentioned that there are two criteria to decide whether an assignment is good or not.

First criteria is the number of secondary variables necessary to represent the different states of the machine. Second criteria is the simplicity of the machine, i.e., the number of logical devices required to realize the flow table.

The question that arises is whether these two criteria always agree and, if they contradict each other, to make the optimal assignment.

Ex. 5.7

Given the following flow table:

	N. S.				
P. S.	00	01	10	11	Z
a	b	c	d	a	0
b	a	c	d	a	1
c	b	a	d	a	1
d	b	c	a	a	1

Let us make two different secondary assignments. Assignment α using two secondary variables, assignment β using three secondary variables.

	y_1	y_2
$a \longrightarrow$	0	0
$b \longrightarrow$	0	1
$c \longrightarrow$	1	0
$d \longrightarrow$	1	1

assignment α

	y_1	y_2	y_3
$a \longrightarrow$	0	0	0
$b \longrightarrow$	0	0	1
$c \longrightarrow$	0	1	0
$d \longrightarrow$	1	0	0

assignment β

$y_1 y_2 \backslash x_1 x_2$	00	01	10	11
0 0	01	10	11	00
0 1	00	10	11	00
1 0	01	00	11	00
1 1	01	10	00	00

Excitation matrix - α

$$Y_1 = \bar{y}_1(\bar{x}_1 x_2 + x_1 \bar{x}_2) + y_2 \bar{x}_1 x_2 + \bar{y}_2 x_1 \bar{x}_2$$

$$Y_2 = \bar{x}_1 \bar{x}_2 y_1 + \bar{y}_2 \bar{x}_2 + \bar{y}_1 x_1 \bar{x}_2$$

$$Z = y_1 + y_2$$

And in the same way we obtain from assignment β :

$$Y_1 = \bar{y}_1 x_1 \bar{x}_2$$

$$Y_2 = \bar{y}_2 \bar{x}_1 x_2$$

$$Y_3 = \bar{y}_3 \bar{x}_1 \bar{x}_2$$

$$Z = y_1 + y_2 + y_3$$

As a conclusion from this example we find that the two criteria for optimal assignment may contradict (number of logical devices required in assignment α is larger than in assignment β), hence a method is needed to check in every case which assignment is better and how to find the best one.

Even though the following technique (based on references 13-16) helps in finding a "reasonable" assignment, it is far from being a method of assigning the secondary variable in the optimal way.

If we study assignment β in Ex. 5.7 and assignment b in Ex. 5.1, we notice that in both examples Y_1 depends only on y_1 and the inputs, and Y_2 Y_3 depend only on y_2 y_3 and the inputs.

The fundamental idea in the above mentioned references is to find a method for the selection of these assignments in which each binary variable describing the new state depends on as few variables of the old state as possible. In a later paper Hartmanis[15] expanded the aim of his original research and tried to distinguish between the part of the sequential machine which can be input independent (an outonomous clock as Hartmanis calls it) and the input dependent part.

Hartmanis' s main tool in doing it is the partition with, or without, the substitution property on the set of states of sequential machines.

In his work Hartmanis assumed that the outputs depend only on the present states and not on the inputs. Before studying the method, it is worthwhile to mention that Hartmanis did not prove that his method leads to a unique minimal solution. On the contrary, for some machines better solutions exist, and even though he reduced the dependence between the new states and the old ones, the number of possible assignments still remains very large.

As a final note it must be pointed out that the method is applicable to a restricted class of machines which have several required properties.

The Algebra of the Partitions

In order to have a good understanding of Hartmanis' s method we shall first study the algebraic properties of the partitions and develop the switching theory for them.

Let $I = \{I_1\ I_2 \dots\ I_m\}$ be a finite set of inputs.

$\{S_1 \dots\ S_n\}$ be the set of internal states of a finite set sequential

machine, $\{Z_1 \dots\ Z_n\}$ the corresponding set of outputs.

Let $\{Y_1 \ldots\ldots Y_s\}$ be a set of binary variables describing the internal state of the machine, then the ith variable Y_i of the next state will depend - in general - on I and all the variables describing the old state.

$$Y_i = f(y_1\, y_2 \ldots\ldots y_s : I)$$

We want to find an assignment for which the subset $\{Y_1\, Y_2 \ldots\ldots Y_k\}\ 1 \leq k < s$ can be computed without the knowledge of the remaining state variables.

Hence to make

$$Y_i = f(y_1\, y_2 \ldots y_k : I)$$

Those assignments we call assignments with self dependent subsets. It is not always possible to find for a machine any assignment with self dependent subsets of variables, but there still exist a possibility to decrease the dependence.

For example when $\{Y_1 \ldots\ldots Y_k\}$ depends only on I and $\{\underline{y_r} \ldots\ldots y_s\}$.

<u>Definition:</u>

A Partition π on a set S is defined as a collection of disjoint subsets of S such that their set union is S. If all the subsets of S forming the partition π have the same number of elements, the partition is called uniform. The subsets will be called the blocks of π.

The two trivial partitions are:

$\pi = 0$ Each block consists of a single element.

$\pi = I$ All elements are contained in one block.

It is said that $\pi_1 \geq \pi_2$ (π_1 is larger than, or equal to π_2), if and only if each block of π_2 is contained in a block of π_1.

From this definition we see that there are pairs of partitions that cannot be compared. For such cases we define the least upper bound (l. u. b.) and the greatest lower bound (g. l. b.). The l. u. b. is defined for the pair of partitions π_1 and π_2 as the partition π_3 that satisfies these equations:

If	$\pi_1 \leq \pi_3$	$\pi_2 \leq \pi_3$
and	$\pi_1 \leq \pi_4$	$\pi_2 \leq \pi_4$
then	$\pi_3 \leq \pi_4$	

Hence π_3 is a partition the blocks of which contain every block of π_1 and π_2 and is is smaller than any other partition that has this property.

In the same manner we define π_3 as the g. l. b. if and only if it satisfies the following equations:

If $\quad \pi_3 \leq \pi_1 \qquad\qquad \pi_3 \leq \pi_2$

and $\quad \pi_4 \leq \pi_1 \qquad\qquad \pi_4 \leq \pi_2$

then $\quad \pi_3 \geq \pi_4$

The l. u. b. is denoted by $\pi_1 + \pi_2$ and the g. l. b. by $\pi_1 \cdot \pi_2$.

$\pi_1 \cdot \pi_2$ is a partition the blocks of which are obtained by intersecting the blocks of π_1 and π_2. $\pi_1 + \pi_2$ is a partition the blocks of which consist of all blocks of π_1 and π_2 which are chain connected, i.e.

<u>Example 5. 8</u>

Let S consist of:

$$\{s_1 \; s_2 \cdots\cdots \; s_{11}\}$$

and let
$$\pi_1 = \{\overline{s_1 \, s_2 \, s_3} : \overline{s_4 \, s_5 \, s_6 \, s_7} : \overline{s_8 \, s_9} : \overline{s_{10} \, s_{11}}\}$$
$$\pi_2 = \{\overline{s_1 \, s_2} : \overline{s_3 \, s_4 \, s_5} : \overline{s_6 \, s_7} : \overline{s_8 \, s_9 \, s_{10}} : \overline{s_{11}}\}$$

Then:
$$\pi_1 \cdot \pi_2 = \{\overline{s_1 \, s_2} : \overline{s_3} : \overline{s_4 \, s_5} : \overline{s_6 \, s_7} : \overline{s_8 \, s_9} : \overline{s_{10}}\,\overline{s_{11}}\}$$
$$\pi_1 + \pi_2 = \{\overline{s_1 \, s_2 \, s_3 \, s_4 \, s_5 \, s_6 \, s_7} : \overline{s_8 \, s_9 \, s_{10} \, s_{11}}\}$$

$\pi_1 + \pi_2$ is the l. u. b. or the smallest partition that is larger than π_1 and π_2. $\pi_1 \cdot \pi_2$ is the g.l.b. or the largest partition the blocks of which are contained in the blocks of π_1 and π_2.

The complement π_2 of a partition π_1 is defined as the partition that satisfies these two equations:

$$\pi_1 \cdot \pi_2 = 0$$
$$\pi_1 + \pi_2 = I$$

The partitions satisfy the following laws:

1. $\pi_1 \cdot \pi_2 = \pi_2 \pi_1$ $\qquad$ $\pi_1 + \pi_2 = \pi_2 + \pi_1$

2. $\pi_1 \cdot \pi_1 = \pi_1$ $\qquad$ $\pi_1 + \pi_1 = \pi_1$

3. $\pi_1 \cdot (\pi_2 \cdot \pi_3) = (\pi_1 \cdot \pi_2)\ \pi_3$

   $\pi_1 + (\pi_2 + \pi_3) = (\pi_1 + \pi_2) + \pi_3$

4. $\pi_1 \cdot (\pi_1 + \pi_2) = \pi_1$

   $\pi_1 + (\pi_1 \cdot \pi_2) = \pi_1$

But note that:

5. $\pi_1 \cdot (\pi_2 + \pi_3) \geq \pi_1 \cdot \pi_2 + \pi_1 \cdot \pi_3$

   $\pi_1 + \left[\pi_2 \cdot \pi_3\right] \leq (\pi_1 + \pi_2) \cdot (\pi_1 + \pi_3)$

<u>Definition:</u>

A partition π on a set of states of sequential machine M is said to have the substitution property if for any two states S_i and S_j belonging to the same block of π and any input I the states IS_i and IS_j are again contained in a common block of π. IS_i is the next state the machine goes into from S_i when the input I is applied.

If we consider ex. 5.1. we find that the partitions are:

$$\pi_1 = \left\{ \overline{0, 1, 2} \ : \ \overline{3, 4, 5} \right\}$$

$$\pi_2 = \left\{ \overline{0, 5} \ : \ \overline{1, 4} \ : \ \overline{2, 3} \right\}$$

These partitions have the substitution property (S. P.)

The last algebraic property which we shall mention is as follows:

Let π_1 and π_2 be partitions - on a set of states S - with S. P. then the following partitions have also S. P.

1) $\pi_1 \cdot \pi_2$

2) $\pi_1 + \pi_2$

For any set of states S partitions π_i can be found. However these partitions do not necessarily have the S. P. The partition π with S. P. exists for the set of states S if the first k variables $1 \leq k < s$ designating the next state, can be computed from the input and the first k variables of the old state.

Secondary Assignment Using Partitions

We let $\# \pi$ be the number of the blocks in the partition π, and $m(\pi)$ be the number of states in the largest block. Then $k = \log_2 \# \pi$ digits are needed to distinguish between the blocks of π and $\log_2 m(\pi)$ digits are needed to distinguish between the states in the blocks.

In theorem 2 in reference 14 Hartmanis states that if M is a sequential machine with n states, then the existance of a non trivial partition with S. P. - π implies that there exists an assignment of $S = \log_2 n$ binary digits to the states of M such that the first k digits $1 \leq k < S$ of the next state can be computed without the knowledge of the last S-k digits provided

$$S = \log_2 \# \pi + \log_2 m(\pi)$$

The following example will show that these are necessary conditions but they are not sufficient.

The existance of a non trivial partition with S. P. does not assure an assignment of length $S = \log_2 n$.

Ex. 5. 9

	I_1	I_2	I_3	I_4
a	a	b	d	c
b	a	b	b	b
c	a	c	c	c
d	a	c	d	b

I_1	$\longrightarrow 00$
I_2	$\longrightarrow 01$
I_3	$\longrightarrow 11$
I_4	$\longrightarrow 10$

$\pi = (\overline{a, d} \; ; \; \overline{b, c})$ has S. P. hence, the assignment will be:

a $\longrightarrow$ 00

b $\longrightarrow$ 10

c $\longrightarrow$ 11

d $\longrightarrow$ 01

$\# \pi = 2$ $\qquad\qquad$ $m(\pi) = 2$

$$S = \log_2 2 + \log_2 2 = 2$$

Y_1 Y_2	$X_1 X_2$ 0 0	0 1	1 1	1 0
0 0	0 0	1 0	0 1	1 1
1 0	0 0	1 0	1 0	1 0
1 1	0 0	1 1	1 1	1 1
0 1	0 0	1 1	0 1	1 0

$$Y_1 = \bar{X}_1 X_2 + \bar{X}_2 X_1 + X_1 y_1$$
$$Y_2 = X_2 y_2 + \bar{y}_1 \bar{y}_2 X_1 + y_1 y_2 X_1$$

Even though it seems that a solution was found for the secondary variables, this flow table cannot be realized with two secondary variables because it includes several races and critical races. Hence Theorem 2 holds only for synchronous circuits and for asynchronous circuits provided that there are no races in the flow table.

Furthermore this method does not give a unique solution nor a minimal one.

Ex. 5.10

Consider the flow table of the previous example with the following secondary assignment

$$\pi = (\overline{a, d} : \overline{b, c})$$

$a \longrightarrow 10$
$b \longrightarrow 00$
$c \longrightarrow 01$
$d \longrightarrow 11$

Then:

$$Y_1 = \bar{X}_1 \bar{X}_2 + X_1 X_2 y_1$$
$$Y_2 = X_2 y_2 + X_1 y_1 \bar{y}_2 + X_1 \bar{y}_1 y_2$$

In this solution Y_1 is simpler than in the previous assignment. Considering now the third assignment as follows:

$a \longrightarrow 01$
$b \longrightarrow 11$
$c \longrightarrow 10$
$d \longrightarrow 00$

$$Y_1 = \bar{X}_1 X_2 + X_1 \bar{X}_2 + X_1 y_1$$

$$Y_2 = \bar{X}_1 \bar{X}_2 + \bar{y}_1 \bar{y}_2 \bar{X}_2 + y_1 y_2 + \bar{X}_1 y_2$$

For this assignment Y_2 is more complicated than in the two previous assignments.

Even though the last example was a very simple one with only one partition, the number of different solutions is large and there are no criteria to determine which is the best assignment.

When the flow tables are more complicated, one has to check $\frac{n(n-1)}{2}$ pairs of states in order to find the partitions with S. P. For each partition several assignments are possible, so that the number of possible assignments is still very high. The detection of assignments with more than one self-dependent subset involves consideration of several partitions with S. P. and their mutual relations.

Ex. 5.11.

Given the following flow table

P. S.	I_1	I_2
0	3	6
1	2	7
2	4	5
3	5	5
4	7	2
5	6	3
6	0	1
7	1	1

$$\pi_1 = \{\overline{0,1} : \overline{2,3} : \overline{4,5} : \overline{6,7}\}$$

Y_1 Y_2 are needed in order to distinguish between the blocks. Y_3 distinguishes between the states in the blocks.

But as was shown in reference 14 this assignment does not necessarily give the best solution. If we consider the partition

$$\pi_2 = \{\overline{0,4} : \overline{1,5} : \overline{2,6} : \overline{3,7}\}$$

we may get a better solution.

Noticing that the partition π_3 is zero for:

$$\pi_3 = \pi_1 \cdot \pi_2 = 0$$

Hence we can assign Y_1 Y_2 to partition π_1 and Y_3 Y_4 to π_2 and we get two self-dependent sets which may result in a simple solution even though the number of secondary variables is 4 instead of 3 in the previous assignments.

But $$\pi_4 = \pi_1 + \pi_2 = \left\{ \overline{0, 1, 4, 5} ; \overline{2, 3, 6, 7} \right\}$$

then Y_1 will distinguish between the blocks of π_4 and Y_2 Y_3 between the states.

Again noticing that π_1 and π_2 each splits the blocks of π_4 to two parts, therefore, we assign Y_1 to distinguish between the blocks of π_4.

Y_1 & Y_2 are assigned to distinguish between the block of π_1 and Y_1 & Y_3 between the blocks of π_2.

The last assignment is:

0 $\longrightarrow$ 001
1 $\longrightarrow$ 000
2 $\longrightarrow$ 101
3 $\longrightarrow$ 100
4 $\longrightarrow$ 011
5 $\longrightarrow$ 010
6 $\longrightarrow$ 111
7 $\longrightarrow$ 110

and the equation of the secondary variables are:

$$Y_1 = \bar{y}_1$$

$$Y_2 = y_1 \bar{y}_2 + \bar{y}_1 y_2 \bar{X} + \overline{y_2}\, X$$

$$Y_3 = \bar{y}_1 y_3 X + \bar{y}_1 \bar{y}_3 \bar{X} + y_1 y_3 \bar{X}$$

In this example we saw that the number of possible assignments is very large and there are no ways to determine whether the solution we have found is the simplest one or simpler solutions can be found.

The above solution was given by Hartmanis in reference 14, but in order to point out on the fact that this method does not give the best solution and does not indicate whether there is a better solution, we shall show that the same machine can be realized with less logical devices when the assignment of the secondary variables is as follows:

$$\pi_4 = \{ \overline{0, 1, 4, 5} : \overline{2, 3, 6, 7} \}$$

Let Y_1 distinguish between the blocks of π_4, and let Y_2 & Y_3 distinguish among the states of the blocks.

	Y_1	Y_2	Y_3	$I_1 = \bar{X}$			$I_2 = X$		
$0 \longrightarrow$	0	0	0	1	0	1	1	1	1
$1 \longrightarrow$	0	0	1	1	0	0	1	1	0
$2 \longrightarrow$	1	0	0	0	1	1	0	1	0
$3 \longrightarrow$	1	0	1	0	1	0	0	1	0
$4 \longrightarrow$	0	1	1	1	1	0	1	0	0
$5 \longrightarrow$	0	1	0	1	1	1	1	0	1
$6 \longrightarrow$	1	1	1	0	0	0	0	0	1
$7 \longrightarrow$	1	1	0	0	0	1	0	0	1

$$Y_1 = \bar{y}_1$$

$$Y_2 = \bar{y}_2 X + y_1 \bar{y}_2 + \bar{y}_1 y_2 \bar{X}$$

$$Y_3 = y_2 \bar{y}_3 + \bar{y}_1 \bar{y}_3 + \bar{y}_3 \bar{X}$$

Y_3 is much simpler in this solution than in the previous one. However there is no assurance that this is the simplest solution, on the contrary it seems that Y_2 & Y_3 can be further simplified.

Decomposition of Sequential Machines

The importance of this subject has already been pointed out by Moore[4]. The advantages of such a decomposition are numerous. It is less complicated to design several small machines than one large machine. The speed of operation of several parallel small machines is higher than that of a large machine.

However, even though the decomposition, when realizable, simplifies the analysis, design, and installation of the machine, it is not always more economical. The decomposition of the sequential machine can be divided into cascaded finite state machines or into parallel machines, in this chapter we shall concern only the latter one.

Hartmanis derived in his papers conditions on two partitions with S. P. under which there exist assignments such that the S binary digits can be split into two parts such that each part will be computed independently.

Let M be a sequential machine with n states and a binary assignment of length $S = \log_2 n$. The assignment S can be split into two parts such that the first K variables $1 \leq K < S$ and the last S - K variables can be computed independently if and only if there exist two nontrivial partitions π_1 and π_2 with S. P. and the following conditions are satisfied:

1) The blocks of π_1 have at most one element in common with any of the blocks of π_2 .

$$\pi_1 \cdot \pi_2 = 0$$

2) $$\log_2 \# \pi_1 + \log_2 \# \pi_2 = S$$

As an example we can use example 5.1. It was shown that assignment b was more economical than assignment a. The partitions for the flow table 5.1 are:

$$\pi_1 = (\overline{0, 1, 2} : \overline{3, 4, 5})$$

$$\pi_2 = (\overline{0, 5} : \overline{1, 4}, : \overline{2, 3})$$

and:

$$\pi_1 \cdot \pi_2 = (\overline{0}, \overline{1}, \overline{2}, \overline{3}, \overline{4}, \overline{5}) = 0$$

These conditions are necessary and sufficient for the decomposition of sequential machines. However Harmanis does nct mention the fact that the decomposition by itself has no value. The decomposition has the advantage of simplifying the machine, but in several cases where condition No. 1 is satisfied but condition 2 is not satisfied the machine becomes more complicated.

As an example to illustrate this point let us consider the flow table in the previous example No. 5.11.

$$\pi_1 = \{\overline{0, 1} : \overline{2, 3} : \overline{4, 5} : \overline{6, 7}\}$$

$$\pi_2 = \{\overline{0, 4} : \overline{1, 5} : \overline{2, 6} : \overline{3, 7}\}$$

$\pi_1 \cdot \pi_2 = 0$ hence condition No. 1 is satisfied.

$$\log_2 \# \pi_1 + \log_2 \# \pi_2 = 4$$

but $4 > \log_2 n$

as $n = 8$ and $\log_2 8 = 3$

The assignment of $Y_1\ Y_2\ Y_3\ Y_4$ may be as follows:

	Y_1	Y_2	Y_3	Y_4
0	0	0	0	0
1	0	0	0	1
2	0	1	1	1
3	0	1	1	0
4	1	1	0	0
5	1	1	0	1
6	1	0	1	1
7	1	0	1	0

Which results in the following equations:

$$Y_1 = \bar{y}_1\ X + y_2\ \bar{X}$$

$$Y_2 = \bar{y}_1\ \bar{X} + y_2\ X$$

$$Y_3 = \bar{y}_3$$

$$Y_4 = y_3\,\bar{y}_4 + \bar{X}\,\bar{y}_3\,y_4 + X\,\bar{y}_3\,\bar{y}_4 + X\,y_3\,y_4$$

$$= y_3\,\bar{y}_4 + \bar{X}\,\bar{y}_3\,y_4 + X\,\bar{y}_4 + X\,y_3$$

This circuit has more diodes than the one designed in the previous example. However, in example 5. 7 we showed that by increasing the number of the secondary elements the circuit was simplified, hence from that example and the last one we can conclude that the decomposition of the sequential machine does not assure a simpler machine. Instead of one machine we' ll need several simpler machines, the total of which, however, may be more complicated than the one larger machine.

In the previous pages necessary and sufficient condition for the decomposition of sequential machines were derived. In reference 16, Hartmanis showed that to every (loop free) realization of a sequential machine from n smaller machines corresponds a set of n partition with S. P. whose product is the zero partition. In order to discuss this subject we shall define as follows:

Let $\{M_1\ M_2 \ldots M_n\}$ be a set of machines in which the outputs of any machine

M_i may be used as inputs to other machines. We shall say that the machine M_i is a predecessor of M_j if an output of M_i is an input of M_j. Any subset of these machines $\{M_{i1}\ M_{i2} \ldots\ M_{ir}\}$ forms a loop in the connection, if the outputs of M_{ik} is an input to M_{ik+1} for $k = 1, 2 \ldots r-1$ and the outpot of M_{ir} is an input to M_{i1}.

In our discussion we concern ourselves with loop-free machines, i.e., sets of machines in which no subset forms a loop in their connection.

Hartmanis proved that to every closed subset C_j of the set of loop-free machines $\{M_1\ M_2 \ldots\ M_n\}$ corresponds a partition π_j with S.P. which is induced by placing in the same block of π_j all states of M which correspond to the same set of states of the machine in C_j. We shall write

$$C_i \subseteq C_j$$

if the set C_j contains at least all the machines in C_i. The conditions for the existance of this inequality are:

$$\pi_i \geq \pi_j$$

Given the set of sequential machines

$$M = \{M_1\ M_2 \ldots\ M_n\}.$$

To every subset C_i of M corresponds a partition with S.P. - π_j provided the set of these partitions,

$$\{\pi_1\ \pi_2 \ldots\ \pi_n\}$$

is such that

$$\prod_{i=1}^{n} \pi_i = 0$$

And conversely, given the set of partitions with S. P.

$$\{\pi_1\ \pi_2 \ldots\ \pi_n\}$$

such that

$$\prod_{i=1}^{n} \pi_i = 0$$

then M can be realized from n machines $\{M_1\ M_2 \ldots\ M_n\}$.

In such a realization the information flows from M_i to M_j if

$$\pi_i \geq \pi_j$$

We shall illustrate this method of realization by the following example.

Ex. 5.12

Given machine M described by the following flow table:

P. S.	N. S.			
	I_1	I_2	I_3	I_4
1	4	2	3	1
2	3	1	4	2
3	2	4	1	3
4	1	3	2	4

For this set of states exist the following set of partitions with S. P. :

$$\pi_1 = \{\overline{1},\ \overline{2},\ \overline{3},\ \overline{4}\} = 0$$

$$\pi_2 = \{\overline{1,\ 2}\ :\ \overline{3,\ 4}\}$$

$$\pi_3 = \{\overline{1,\ 3}\ :\ \overline{2,\ 4}\}$$

$$\pi_4 = \{\overline{1,\ 4}\ :\ \overline{2,\ 3}\}$$

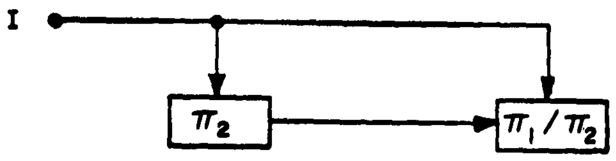
$$\pi_5 = \{\overline{1,\ 2,\ 3,\ 4}\} = I$$

We check now for the subsets which satisfy the condition

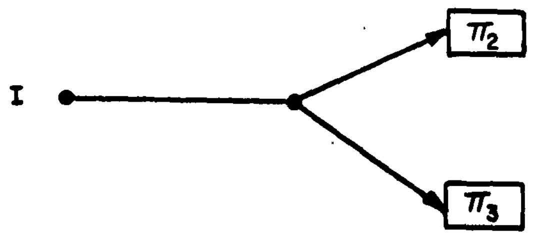
$$\prod_{i=1}^{n} \pi_i = 0$$

1. $\{\pi_1 , \pi_2\}$
2. $\{\pi_1 , \pi_3\}$
3. $\{\pi_1 , \pi_4\}$
4. $\{\pi_2 , \pi_3\}$
5. $\{\pi_2 , \pi_4\}$
6. $\{\pi_3 , \pi_4\}$
7. $\{\pi_1 , \pi_5\}$

The realizations of the first and the fourth sets of partitions are as follows: (The arrows indicate the direction of the information-flow).



realization of set No. 1



realization of set No. 4

(π_1/π_2 means that π_1 is computed from the information given by π_2)

The realization of M from set No. 4 is simple. Two machines M_1 & M_2 - each consisting of one memory element and two states - are operating in parallel.

In the realization of set No. 1 M consists of two machines M_1 and M_2. Since $\pi_2 \geq \pi_1$, we get a cascade of M_1 & M_2 in which the output of M_2 is the input of M_1. From π_2 we obtain that M_2 has two states $\alpha = \overline{1,2}$ $\beta = \overline{3,4}$ corresponding to the blocks of π_2 .

P. S.	I_1	I_2	I_3	I_4
α	β	α	β	α
β	α	β	α	β

Flow table for M_2

However as $I_1 \equiv I_3$ and $I_2 \equiv I_4$ we obtain:

P. S.	I_1	I_2
α	β	α
β	α	β

Reduced flow table for M_2

Where $I_1 = I_1 \vee I_3$

$I_2 = I_2 \vee I_4$

M_1 is constructed from the partition τ such that $\pi_2 \cdot \tau = 0$.
Since π_2 has 2 blocks τ must have at least 2 blocks.

We choose $\tau = \{\overline{14} : \overline{23}\} = \{\gamma : \delta\}$

The resulting machine has a input, the exterior inputs I and additional inputs α or β, designating the P. S. of M_2.

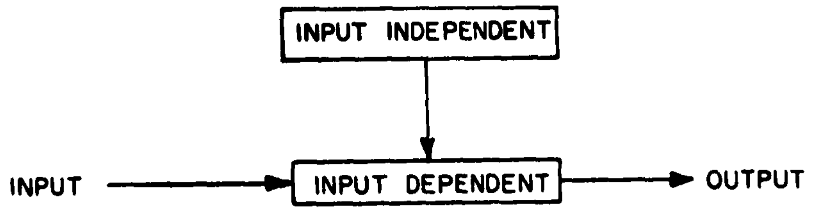
P. S.	αI_1	βI_1	αI_2	βI_2
γ	γ	γ	δ	δ
δ	δ	δ	γ	γ

$$I_1 \equiv I_1 \vee I_4$$
$$I_2 \equiv I_2 \vee I_3$$

Reduced flow table for M_1

D. Input Independent Sequential Machines

This chapter deals with a certain type of sequential machines which can be divided into two smaller machines, an input independent machine and an input dependent one. Hartmanis[15] found some of the necessary and sufficient conditions for the existance of the input-independent part in the machine (the "autonomous clock"). The realization of the machine from an input independent and an input dependent machine is as follows:



The main tool in this study is the partition with the S. P. But in order to determine the input independent variables in addition to the S. P., the partition π has to be so chosen that it is larger than or equal to the partition π_I which is induced by the identification of the inputs.

$$\pi \geq \pi_I$$

Furthermore we want to determine the conditions for the existance of such an assignment of S variables $Y_1 \dots . Y_s$ such that the first k variables $1 \leq k \leq s$ do not depend 1) on the input 2) on the last $s - k$ variables.

It is worthwhile to mention that the additional requirement may result in a more complicated solution than the one that could have been found without the input independent variables. Furthermore, there is no unique input independent machine and Hartmanis did not find any criteria to decide which is the largest input independent machine. This method makes the search for the assignment very tedious and it does not promise any better results. However, as the method offers new ideas in the study of the secondary assignment problem, it will be briefly given in this chapter.

In order to compute π_I we have to identify all next states which are contained in the same present state of the flow table, and then add all identifications of states which can be obtained by the transitive law.

<u>Ex. 5.13.</u>

Given the following flow table

	N. S.	
P. S.	X = 0	X = 1
0	5	4
1	4	4
2	1	0
3	0	1
4	7	7
5	6	7
6	3	2
7	3	3

Computation of π_I :

$5 = 4$
$0 = 1$
$6 = 7$
$2 = 3$

Hence: $\pi_I = \{\overline{0, 1} : \overline{2, 3} : \overline{4, 5} : \overline{6, 7}\}$

A check shows that π_I has S. P. Y_1 Y_2 will distinguish between the blocks of π_I . Y_3 between the states of the blocks.

$0 \longrightarrow 0\ 0\ 0$
$1 \longrightarrow 0\ 0\ 1$
$2 \longrightarrow 0\ 1\ 0$
$3 \longrightarrow 0\ 1\ 1$
$4 \longrightarrow 1\ 0\ 0$
$5 \longrightarrow 1\ 0\ 1$
$6 \longrightarrow 1\ 1\ 0$
$7 \longrightarrow 1\ 1\ 1$

The corresponding equations are:

$$Y_1 = \overline{y}_2$$
$$Y_2 = y_1$$
$$Y_3 = \overline{y}_3 \overline{X} + y_2 y_3 X + y_1 y_2 y_3 + y_1 \overline{y}_2 X$$

As seen from these equations Y_1 & Y_2 are input independent where Y_3 is input dependent. However, the partition $\pi = \{ \overline{0,1,6,7} : \overline{2,3,4,5} \}$ satisfies the requirement of S. P. and $\pi > \pi_I$; therefore, it is another autonomous clock.

<u>Ex. 5.14.</u>

Consider the following flow table:

P. S.	N. S.	
	X = 0	X = 1
0	5	3
1	4	2
2	1	5
3	0	4
4	3	1
5	2	0

$$\pi_I = \{ \overline{0, 2, 4} : \overline{1, 3, 5} \}$$

π_I has S. P.

Let us check several possible assignment for this flow table:

<u>assignment a</u>

$0 \longrightarrow 0\ 0\ 0$
$1 \longrightarrow 1\ 0\ 0$
$2 \longrightarrow 0\ 0\ 1$
$3 \longrightarrow 1\ 0\ 1$
$4 \longrightarrow 0\ 1\ 1$
$5 \longrightarrow 1\ 1\ 1$

The corresponding equations are:

$$Y_1 = \overline{y}_1$$
$$Y_2 = \overline{y}_3 \overline{X} + \overline{y}_2 y_3 X$$
$$Y_3 = \overline{y}_2 X + \overline{y}_3 \overline{X} + y_2 \overline{X}$$

<u>assignment β</u>

In order to improve the results of assignment α, one can see that to distinguish between the states in each block we used Y_2 & Y_3 in the following manner:

00 01 11

But with this assignment we used Y_2 & Y_3 together when only Y_2 could do the same thing; hence we shall assign Y_2 & Y_3 as follows:

00 01 10

		$\bar{X}$	X
0 $\longrightarrow$	0 0 0	1 1 0	1 0 1
1 $\longrightarrow$	1 0 0	0 1 0	0 0 1
2 $\longrightarrow$	0 0 1	1 0 0	1 1 0
3 $\longrightarrow$	1 0 1	0 0 0	0 1 0
4 $\longrightarrow$	0 1 0	1 0 1	1 0 0
5 $\longrightarrow$	1 1 0	0 0 1	0 0 0
6 $\longrightarrow$	0 1 1	-	-
7 $\longrightarrow$	1 1 1	-	-

$$Y_1 = \bar{y}_1$$

$$Y_2 = \bar{y}_2 \bar{y}_3 \bar{X} + y_3 X$$

$$Y_3 = y_2 \bar{X} + \bar{y}_2 \bar{y}_3 X$$

However the partition

$$\pi = \{\overline{0,1} : \overline{2,3} : \overline{4,5}\}$$ has S. P.

hence may result in a better solution even though it is not input independent.

<u>assignment γ</u>

Y_1 & Y_2 distinguish between the blocks. Y_3 between the states.

0 $\longrightarrow$	0 0 0
1 $\longrightarrow$	0 0 1
2 $\longrightarrow$	0 1 0
3 $\longrightarrow$	0 1 1
4 $\longrightarrow$	1 0 0
5 $\longrightarrow$	1 0 1

$$Y_1 = \bar{y}_1 \bar{y}_2 \bar{X} + Y_2 X$$
$$Y_2 = y_1 \bar{X} + \bar{y}_1 \bar{y}_2 X$$
$$Y_3 = \bar{y}_3$$

This assignment requires the same number of diodes as assignment β ; hence the fact that assignment β was in part input independent did not contribute to the simplification of the machine.

E. Assignments using partitions without S. P.

In the previous examples it was possible to find non-trivial partitions with S. P., and using these partitions we could make an economical secondary assignment. However, in many practical problems it is impossible to find for the machine a partition with S. P. . The second fact is that the use of the partition with the S. P. is not always the best approach to assign the secondary variables.

Hartmanis knew these problems, and in reference 14 II he tried to develop a new technique which covers some of these problems. However, the technique has several limitations which prevent it from being usable for most proctical cases.

Hartmanis defines a Partition Pair (π : π') as a pair of partitions such that if S_i and S_j both belong to the same block of π , then for each input I , IS_i and IS_j are in the same block of π' . (IS_i is the state the machine goes into from S_i when I is applied) .

From this definition it follows that if we know in which block of π_1 the state of the machine is contained (which is given by the value of Y_1), then for any input we can compute the block of π_2 in which the next state of the machine will be contained.

In the special case where

$$\pi = \pi'$$

the partition has S. P.

Ex. 5.15

In order to show the advantage and the necessity of this technique consider the following flow table:

P. S.	N. S.				Z
	I_1	I_2	I_3	I_4	
1	1	2	3	4	1
2	3	4	1	2	1
3	2	1	4	3	0
4	4	3	2	1	0

$$\pi_1 = \{\overline{1, 4} \;:\; \overline{2, 3}\} \quad \text{has} \quad \text{S. P.}$$

Assigning Y_1 to distinguish between the blocks and Y_2 between the states we get:

$1 \longrightarrow 0\,0$		$I_1 \longrightarrow 0\,0$	
$2 \longrightarrow 1\,0$		$I_2 \longrightarrow 1\,0$	
$3 \longrightarrow 1\,1$		$I_3 \longrightarrow 1\,1$	
$4 \longrightarrow 0\,1$		$I_4 \longrightarrow 0\,1$	

assignment α

$$Y_1 = \bar{X}_1 y_1 + X_1 \bar{y}_1$$

$$Y_2 = \bar{X}_2 y_1 \bar{y}_2 + \bar{X}_2 \bar{y}_1 y_2 + X_2 \bar{y}_1 \bar{y}_2 + X_2 y_1 y_2$$

$$Z = \bar{y}_2$$

For the same machine we can find the partition pair without S. P.

$$\pi_2 = \pi(y_1) = \{\overline{1, 2} \;:\; \overline{3, 4}\}$$

$$\pi_3 = \pi(y_2) = \{\overline{1, 3} \;:\; \overline{2, 4}\}$$

$1 \longrightarrow 0\,0$		$I_1 \longrightarrow 0\,0$	
$2 \longrightarrow 0\,1$		$I_2 \longrightarrow 0\,1$	
$3 \longrightarrow 1\,0$		$I_3 \longrightarrow 1\,0$	
$4 \longrightarrow 1\,1$		$I_4 \longrightarrow 1\,1$	

assignment β

For assignment β we find:

$$Y_1 = \bar{X}_1 y_2 + X_1 \bar{y}_2$$

$$Y_2 = \bar{X}_2 y_1 + X_2 \bar{y}_1$$

$$Z = \bar{y}_1$$

Notice that Y_1 depends only on y_2 and Y_2 only on y_1 . The machine is "almost decomposed" into two partial machines.

From example 5.15 we see the importance of the partition pairs that do not have substitution property. However, many times we can get same results from the partition with S. P. and from the partition pairs without S. P. Generally the number of partition pairs is larger than the number of partitions with S. P.; therefore, the number of possible assignments is very large.

Ex. 5.16

For the following flow table Hartmanis found about 12 partitions without S. P. and 4 partition pairs from which, after recognizing that the multiplication of three partitions results in the zero partition (which is not easy in general to see), he assigned the secondary variables. The solution below will show that same result can be achieved by using the partition with the S. P. after two trials (even though Hartmanis [14 II] in pg. 596 disagrees with this possibility) :

P. S.		N. S.		Z
	0 0	0 1	1 1	
1	6	1	3	0
2	5	2	4	0
3	4	1	5	0
4	3	2	6	0
5	2	3	5	1
6	1	4	6	0

The partition with S. P. are:

$$\pi_1 = \{\overline{1, 2} : \overline{3, 4} : \overline{5, 6}\}$$

$$\pi_2 = \{\overline{1, 3, 5} : \overline{2, 4, 6}\}$$

Y_1 is assigned for π_2

Y_2 and Y_3 are assigned for π_1

This assignment is possible because $\pi_1 \cdot \pi_2 = 0$.

$1 \longrightarrow 0\ 0\ 0$
$2 \longrightarrow 1\ 0\ 0$
$3 \longrightarrow 0\ 0\ 1$
$4 \longrightarrow 1\ 0\ 1$
$5 \longrightarrow 0\ 1\ 0$
$6 \longrightarrow 1\ 1\ 0$

assignment α

Y_1	Y_2	Y_3	$\bar{X}_1\ \bar{X}_2$	$\bar{X}_1\ X_2$	$X_1\ X_2$
0	0	0	1 1 0	0 0 0	0 0 1
1	0	0	0 1 0	1 0 0	1 0 1
0	0	1	1 0 1	0 0 0	0 1 0
1	0	1	0 0 1	1 0 0	1 1 0
0	1	0	1 0 0	0 0 1	0 1 0
1	1	0	0 0 0	1 0 1	1 1 0
0	1	1	-	-	-
1	1	1	-	-	-

exitation matrix for assignment α

The corresponding equations for assignment α are:

$$Y_1 = \bar{y}_1 \bar{X}_2 + y_1 X_2$$

$$Y_2 = \bar{X}_2 \bar{y}_2 \bar{y}_3 + X_1 y_2 + X_1 y_3$$

$$Y_3 = \bar{X}_2 y_x + \bar{X}_1 X_2 y_2 + X_1 \bar{y}_2 \bar{y}_3$$

However it can be easily detected that the large terms of Y_2 and Y_3 exist because the number of 1's is small; therefore, we assign the following assignment:

$1 \longrightarrow 0\ 0\ 0$
$2 \longrightarrow 1\ 0\ 0$
$3 \longrightarrow 0\ 1\ 0$
$4 \longrightarrow 1\ 1\ 0$
$5 \longrightarrow 0\ 1\ 1$
$6 \longrightarrow 1\ 1\ 1$

assignment β

The equations relating the variables of the old and new states are:

$$Y_1 = \bar{y}_1 \bar{X}_2 + y_1 X_2$$

$$Y_2 = y_3 \bar{X}_2 + \bar{y}_3 X_2 + X_1$$

$$Y_3 = \bar{y}_2 \bar{X}_2 + y_2 X_1$$

In his papers Hartmanis did not mention the fact that this method using partition pairs without substitution property is not usable when

$\pi_1 \cdot \pi_2 \cdots \pi_n \neq 0$ where $n \geq 2$.

Considering the flow table in example 5.1, the partition pair is:

$$\pi = \{ \overline{0, 1, 3} : \overline{2, 4, 5} \}$$

$$\pi' = \{ \overline{1, 3, 5} : \overline{0, 2, 4} \} .$$

If we let Y_1 distinguish between the blocks of π and Y_2 between the blocks of π', we are left with the necessity to assign Y_3 without any rule.

$$\begin{array}{ccc} & Y_1 & Y_2 \\ 0 \longrightarrow & 0 & 1 \\ 1 \longrightarrow & 0 & 0 \\ 2 \longrightarrow & 1 & 1 \\ 3 \longrightarrow & 0 & 0 \\ 4 \longrightarrow & 1 & 1 \\ 5 \longrightarrow & 1 & 0 \end{array}$$

Y_3 must distinguish between the states 1 & 3 and 2 & 4 .

Using the partition pair ($\tau : \tau'$)

$$\tau = \{ \overline{0 1} : \overline{2, 3} : \overline{4, 5} \}$$

$$\tau' = \{ \overline{0, 2} : \overline{1, 4} : \overline{3, 5} \}$$

where

$$\tau \cdot \tau' = 0$$

results in an assignment using 4 variables which is too much for a six-row flow table. This conclusion can be emphasized better by studying the following theoretical partition pair.

$$\pi = \{ \overline{0, 1, 3} : \overline{2, 4, 5, 6} \}$$

$$\pi' = \{ \overline{0, 2, 4, 6} : \overline{1, 3, 5} \}$$

assigning Y_1 for π and Y_2 for π' we get:

$$\begin{array}{lcc}
 & Y_1 & Y_2 \\
0 \longrightarrow & 0 & 0 \\
1 \longrightarrow & 0 & 1 \\
2 \longrightarrow & 1 & 0 \\
3 \longrightarrow & 0 & 1 \\
4 \longrightarrow & 1 & 0 \\
5 \longrightarrow & 1 & 1 \\
6 \longrightarrow & 1 & 0
\end{array}$$

In order to distinguish between states 2, 4 & 6 we need at least 2 more variables, hence we need 4 secondary variables for a 7 - states flow table which could be realized with only 3 variables.

Remarks

The method presented in the last chapter is the first that put some order in the techniques of assigning the secondary variables. Even though the method is far from being ideal or complete, it gives the designer a direction to start his design. The mathematical procedure does not give a unique solution, nor does it promise the minimal solution. However, the number of possibilities of assigning the secondary variables is reduced, and a fairly simple solution can be achieved after several simple tests.

In flow tables where the number of inputs - I_i - is large, it is impossible to find input independent subsets or to find partitions with S. P., but it seems that any solution to the problem of the secondary assignment will come from this direction.

References

1) S. H. Caldwell, "Switching Circuits and Logical Design" - Wiley - 1958, pp. 453-662.

2) D. A. Huffman, "Synthesis of Sequential Switching Circuits" - Journal Franklin Inst., Vol. 257, pp. 161-190, 275-303, March-April 1954.

3) G. H. Mealy, "A Method of Synthesising Sequential Circuits" - BSTJ. Vol. 34 pp. 1045-1079, Sept. 1955.

4) E. F. Moore, "Gedanken Experiments on Sequential Machines" - Automata Studies, Princeton University Press, pp. 129-153, 1956.

5) W. S. Humphrey, "Switching Circuits" - McGraw-Hill, 1958, pp. 212-254.

6) W. Keister, E. Ritchie, S. Washburn, "The Design of Switching Circuits" - Van Nostrand - 1951, pp. 142-175.

7) D. E. Muller, "Theory of Asynchronous Circuits" - Reports No. 66 (1955), 75 (1956), 78 (1957), Digital Computer Labs, University of Illinois.

8) D. A. Huffman, "The Synthesis of Linear Sequential Coding Networks" - Information Theory - Symposium on information theory held at the Royal Inst. London in 1955. Published in 1956.

9) S. Ginsburg, "A Technique for the Reduction of a Given Machine to a Minimal State Machine" - I. Assoc. Comp. Mach., Vol. 6, pp. 259-282, 346, April -959.

10) M. C. Paull and S. H. Unger, "Minimizing the Number of States in Incompletely Specified Sequential Switching Functions" - PGEC, Vol. EC-8, pp. 356-367, Sept. 1959.

11) E. J. McCluskey Jr. and S. H. Unger, "A Note on the Number of Internal Variable Assignment for Sequential Switching Circuits" - I. R. E. Transaction on Electronic Computers, Vol. EC-8, pp. 439-440, Dec. 1959.

12) S. H. Unger, "Hazards and Delays in Asynchronous Sequential Switching Circuits" - I. R. E. Transactions on Circuit Theory, Vol. CT-6, pp. 12-25, March 1959.

13) J. Hartmanis, "Symbolic Analysis of a Decomposition of Information Processing Machines" - Information and Control, Vol. 3, pp. 153-178, June 1960.

14) J. Hartmanis, "On the State Assignment Problem for Sequential Machines I and II" - I. R. E. Transaction On Electronic Computers, Vol. EC-10, Number 2, June 1961, and Vol. EC-10, Number 4, December 1961.

15) J. Hartmanis, "Maximal Autonomous Clocks of Sequential Machines" - I. R. E. Trans. on Electronic Computers, Feb. 1962.

16) J. Hartmanis, "Loop-Free Structure of Sequential Machines" - G. E. Report No. 61-RL-2878E, Nov. 1961.

17) E. J. McCluskey and T. C. Bartee, "A. Survey of Switching Circuit Theory" - McGraw-Hill, 1962.